PATENT ABSTRACTS OF JAPAN

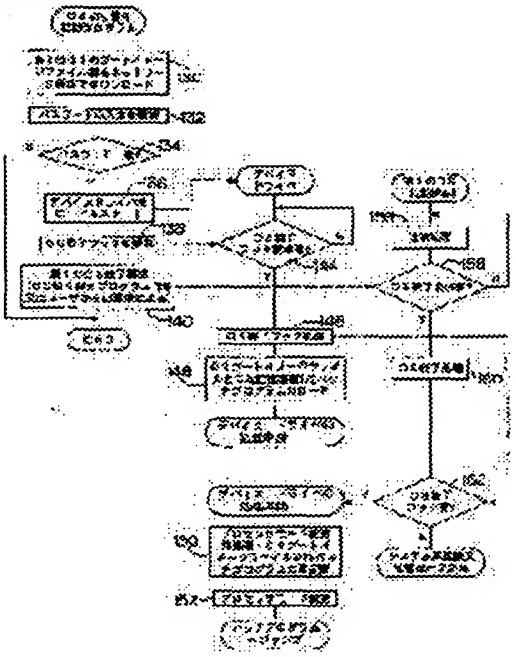
(11)Publication number : 2001-100983
(43)Date of publication of application : 13.04.2001

(51)Int.Cl. G06F 9/06
G06F 9/46

(21)Application number : 11-273860
(22)Date of filing : 28.09.1999
(71)Applicant : INTERNATL BUSINESS MACH CORP <IBM>
(72)Inventor : ITO HIROSHI
KATO NAOTAKA

(54) CONTROL METHOD OF COMPUTER, COMPUTER, AND RECORDING MEDIUM

(57)Abstract:
PROBLEM TO BE SOLVED: To provide a general method of making easily actualizable the switching of an OS to be booted on a computer.
SOLUTION: While a 1st OS is running on a computer, information for booting a 2nd OS is downloaded (130) by browsing homepages or receiving electronic mail, a device driver is actuated after password matching, and a request to hook an OS end is made (134 to 138). The device driver performs a process for hooking the OS end at the request and after the downloaded boot image file, etc., is loaded to a main memory, the process is interrupted (144 to 148). After the OS ends, the process of the device driver is restarted to change the mode of a processor (150, 152) and then a batch program is started. Consequently, the 2nd OS is booted with the boot image file loaded to the main memory after various preprocesses are completed.



LEGAL STATUS

[Date of request for examination]	28.12.1999
[Date of sending the examiner's decision of rejection]	
[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]	
[Date of final disposal for application]	
[Patent number]	3330569
[Date of registration]	19.07.2002
[Number of appeal against examiner's decision of rejection]	
[Date of requesting appeal against examiner's decision of rejection]	
[Date of extinction of right]	

CLAIMS

[Claim(s)]
[Claim 1] The information for making the 2nd different OS from the 1st above OS boot in the state where the 1st OS is working on a computer is acquired under control of the 1st OS. If the information which carried out [aforementioned] acquisition is written in the main storage of the aforementioned computer and the end of the 1st OS is detected The control method of a computer of making the 2nd OS booting on the aforementioned computer using the information written in the aforementioned main storage, without eliminating the information written in the aforementioned main storage.
[Claim 2] The control method of the computer according to claim 1 characterized by acquiring the information for making the 2nd above OS boot by receiving as a file from the exterior of the aforementioned computer through a communication line under control of the 1st above OS.
[Claim 3] The control method of the computer according to claim 2 characterized by once memorizing the information which carried out [aforementioned] acquisition to the secondary storage of the aforementioned computer.
[Claim 4] The information for making the 2nd above OS boot is the control method of the computer according to claim 1 characterized by acquiring the information for being written in the secondary storage of the aforementioned computer in advance, and making the 2nd above OS boot by reading from the aforementioned secondary storage under control of the 1st above OS.
[Claim 5] To the information for making the 2nd above OS boot The boot image file of the 2nd above OS is included, and the boot image file of the 2nd above OS is written in the main storage of the aforementioned computer. By making it the field which wrote in the boot image file of the 2nd above OS of the aforementioned main storage in bootstrap code of a firmware accessed, after detecting the end of the 1st above OS The control method of the computer according to claim 1 characterized by making the 2nd above OS boot on the aforementioned computer.
[Claim 6] The control method of a computer according to claim 5 that the boot image file of the 2nd above OS is characterized by being created for secondary storages.
[Claim 7] The control method of the computer according to claim 5 characterized by performing making it the field which wrote in the boot image file of the 2nd above OS of the aforementioned main storage in bootstrap code of a firmware accessed by hooking a secondary-storage I/O-service code.
[Claim 8] The control method of the computer according to claim 1 characterized by rearranging informational a part or informational all that was written in the account of before to the field which is not overwritten by boot of the 2nd OS at least among the storage regions of the aforementioned main storage once writing the information for making the 2nd above OS boot in the field which is not overwritten under control of the 1st OS among the storage regions of the aforementioned main storage.
[Claim 9] After making the 2nd OS boot on the aforementioned computer, in the state where the 2nd OS is working The program which updates a firmware, the program which performs the network boot which needs a special protocol, The program which sets up, the program which performs a diagnosis of computer system, The control method of the computer according to claim 1 which operation is not guaranteed or is characterized by performing at least one of the program **s which perform processing unsuitable for execution in the working

state of the 1st OS.
[Claim 10] The control method of the computer according to claim 1 characterized by making a series of processings in which it results in boot of the 2nd OS on the aforementioned computer at least perform continuously automatically from boot of the 1st OS on a computer.
[Claim 11] The control method of the computer according to claim 1 characterized by making boot non-performed OS boot on the aforementioned computer among two or more aforementioned sorts of OS's whenever it acquires respectively the information for making two or more sorts of mutually different OS's boot as information for making the 2nd different OS from the 1st above OS boot and detects the end of OS.
[Claim 12] The control method of the computer according to claim 1 characterized by checking behind whether it is OS with 2nd OS just whether the user of a computer is the just user of the 2nd OS on 1st OS before acquiring the information for making the 2nd OS boot.
[Claim 13] The computer characterized by providing the following. An acquisition means to acquire the information for making the 2nd different OS from the 1st above OS boot in the state where the 1st OS is working under control of the 1st OS. Boot control means which make the 2nd OS boot on the aforementioned computer using the information written in the aforementioned main storage, without eliminating the information written in main storage by the aforementioned write-in means if the end of the 1st OS is detected as the write-in means which writes the information acquired by the aforementioned acquisition means in the main storage of the aforementioned computer.
[Claim 14] The 1st step which acquires the information for making the 2nd different OS from the 1st above OS boot in the state where the 1st OS is working on a computer under control of the 1st OS, If the end of the 2nd step and the 1st OS which writes the information which carried out [aforementioned] acquisition in the main storage of the aforementioned computer is detected The record medium with which the program for making a computer perform processing containing the 3rd step which makes the 2nd OS boot on the aforementioned computer using the information written in the aforementioned main storage, without eliminating the information written in the aforementioned main storage was recorded.

TECHNICAL PROBLEM

[Problem(s) to be Solved by the Invention] However, the method of setting FD to FDD has the fault of being inapplicable in the computer in which FDD is not prepared. moreover -- setting to FDD FD which created FD for starting the 2nd OS, or was created **** -- etc. -- work is also complicated and the number of the computers for work increases -- it is alike, and it follows and the complicatedness of work becomes very remarkable
[0006] moreover, by the way a firmware with a built-in network adapter communicates with a server, and acquires information from a server Communication with a firmware and a server Boot Carrying out A premised special bootstrap protocol (for example, RFC951 () which IETF (Internet Engineering Task Force) has specified) [InternetRequest For Comments] Since the bootstrap protocol in 951 etc. is realized a server -- the above -- while supporting the special protocol, acquisition of the information for making it boot has a limit under the influence of network-configuration elements, such as a fire wall and a router, and the problem of being scarce is in versatility
[0007] Moreover, storage regions, such as a hard disk, are divided into two or more partitions as other methods of making the 2nd OS boot. The boot image for making each partition boot a mutually different OS is memorized. After rewriting the master boot record which defined whether it would access and boot to which partition at the during starting of a computer, by starting a computer, the method of making two or more sorts of OS's boot alternatively is also considered.
[0008] However, since the boot image for booting the 2nd OS by this method will reside permanently in storage regions, such as a hard disk Even if it is the case that the operating frequency of the 2nd OS is low, one of two or more partitions will be occupied by the 2nd OS. After the problem that storage regions, such as a hard disk, cannot be used effectively, and the 1st OS are installed, there is a problem that it is not easy to build the

partition for the 2nd OS.
[0009] moreover, by the method of memorizing the boot image of the 2nd OS to a hard disk memorizing collectively the application program which should be performed under control of the 2nd OS — being general (the same being said of the method which FD is made to memorize) -- For example, the renewal of the firmware which is an example of processing which should change and perform OS Since it is necessary to carry out by the newest application program for updating to the newest firmware after upgrade coming to hand whenever there is upgrade of a firmware etc. Even if it stations such an application program at a hard disk permanently, there is almost no merit. Moreover, after performing an application program, eliminating from a hard disk is considered, and there is also a problem that the work to eliminate is complicated. Moreover, when file systems generally differ by the 1st OS and 2nd OS, updating and elimination of the application on 2nd OS partition become difficult from the application on 1st OS.
[0010] It can realize easily that this invention changes OS which constituted in consideration of the above-mentioned fact, and is made to boot on a computer, and aims at acquiring the control method of a computer excellent in versatility.
[0011] Furthermore, it aims at offering the method of changing OS made to boot, without adding change of hardware or a firmware to the existing computer.
[0012] Moreover, this invention aims at acquiring easily OS made to boot for a switchable computer and a switchable record medium.

[0034]
[Embodiments of the Invention] Hereafter, with reference to a drawing, an example of the operation form of this invention is explained in detail. The hardware composition of the computer system 10 which changes from the typical personal computer (PC) suitable for realizing this invention to drawing 1 is typically shown for every subsystem. An example of PC which realizes this invention is notebook type PC12 (refer to drawing 2) which carried "OS/2" of "Windows 98 or NT" or U.S. IBM of U.S. Microsoft Corp. as an operating system (OS) based on OADG (PC Open Architecture Developer's Group) specification. Hereafter, each part of computer system 10 is explained.
[0035] CPU14 which is the brains of the computer system 10 whole performs various programs under control of OS. CPU by CPU-chip"Pentium" made from for example, U.S. Intel, the "MMX technology Pentium", "Pentium Pro", and the other companies, such as AMD, is sufficient as CPU14, and "PowerPC" made from IBM is sufficient as it. CPU14 is storing the code which is accessed frequently and which was restricted very much and data temporarily, and is constituted including the L2(level 2)-cache which is the high-speed operation memory for shortening the total access time to main memory 16. Generally an L2-cache consists of SRAM (static RAM) chips, and the storage capacity is 512kB(s) or more than it.
[0036] The interconnection of CPU14 is carried out to each below-mentioned hardware component through the bus of three hierarchies called I/O bus 22 which consists of FSB18 as a processor direct connection bus directly linked with the own external pin, the PCI (Peripheral Component Interconnect) bus 20 as a high-speed bus for I/O devices, the ISA (Industry Standard Architecture) bus as a low-speed bus for I/O devices, etc.
[0037] FSB18 and PCI bus 20 are connected by the bridge circuit (host-PCI bridge) generally called memory / PCI control chip 24. The memory / PCI control chip 24 of this operation form have composition containing the memory controller ability for controlling access operation to main memory 16, the data buffer for absorbing the difference of the data transfer rate between FSB18 and PCI bus 20, etc., for example, can use 440EX(s), 440GX, etc. made from Intel.
[0038] Main memory 16 is memory which is used as a reading field of the executive program of CPU14 as a working area which writes in the processed data of an executive program and which can be written in. Generally, main memory 16 can consist of two or more DRAM (dynamic RAM) chips, for example, can equip 32MB standardly, and can extend it to 256MB. DRAMs are the high-speed pages DRAM and EDO to meet the demand of improvement in the speed further in recent years. DRAM, a Synchronous DRAM (SDRAM), burst EDO It has

changed to DRAM, RDRAM, etc.

[0039] In addition, firmwares, such as an application program turned to the various device drivers for carrying out hardware operation of OS's, such as Windows 98, and the peripheral devices and specific business and BIOS (Basic Input/Output System : program for controlling the input/output operation of each hardware, such as a keyboard and a floppy disk drive) stored in the flash ROM 56 (it mentions later for details), are contained in an executive program here.

[0040] PCI bus 20 is a type bus (for example, bus width of face of 32/64 bit, maximum frequency of operation 33 / 66/100MHZ, maximum data rate 132 / 264MBps) in which comparatively high-speed data transmission is possible, and the PCI devices like the CardBus controller 30 driven comparatively at high speed are connected to this. In addition, PCI architecture stemmed from proposal of U.S. Intel, and has realized the so-called plug-and-play (plug and play) function.

[0041] The video subsystem 26 is a subsystem for realizing the function relevant to video, and it contains the video controller which reads drawing information from VRAM and is outputted to a liquid crystal display (LCD) 28 (refer to drawing 2) as drawing data while it once writes the drawing information which actually processed the drawing instruction from CPU14, and processed it in video memory (VRAM). Moreover, a video controller can change a digital video signal into the video signal of an analog by the attached digital to analog converter (DAC). The video signal of an analog is outputted to a CRT port (illustration abbreviation) through a signal line.

[0042] Moreover, the CardBus controller 30, the audio subsystem 32, and the modem subsystem 34 are respectively connected to PCI bus 20. The CardBus controller 30 is an exclusive controller for making the bus signal of PCI bus 20 link with the interface connector (CardBus) of the PCI CardBus slot 36 directly. It is arranged in the CardBus slot 36 by the wall surface of for example, PC12 main part, and it is loaded with the PC card (not shown) based on the specification (for example, "PC Card standard 95") upon which PCMCIA (Personal Computer Memory Association)/JEIDA (Japan Electronic Industry Development Association) decided.

[0043] Moreover, communication lines, such as LAN and the telephone line, are connected to the modem subsystem 34. Connection of computer system 10 is enabled through these communication lines at the Internet.

[0044] PCI bus 20 and I/O bus 22 are mutually connected by the multirole PCI device 38. The multirole PCI device 38 is equipped with the bridge function of PCI bus 20 and I/O bus 22, DMA controller ability, a programmable interruption controller (PIC) function and the programmable interval timer (PIT) function, the IDE (Integrated Drive Electronics) interface function, the USB (Universal Serial Bus) function, and the SMB (System Management Bus) interface function, for example, a device called PIIX4 made from Intel can be used for it.

[0045] In addition, DMA controller ability is a function for performing data transfer between a peripheral device (for example, FDD) and main memory 16 without the intervention of CPU14. Moreover, a PIC function is a function to answer an interrupt request (IRQ) from a peripheral device, and to perform a predetermined program (interrupt handler). Moreover, a PIT function is a function to generate a timer signal a predetermined period, and the generating period is programmable.

[0046] Moreover, the IDE hard disk drive (HDD) 40 is connected to the IDE interface realized by the IDE interface function, and also ATAPI (AT Attachment Packet Interface) connection of the IDECD-ROM drive 42 is made. Moreover, IDE The IDE equipment type [other] like a DVD (Digital Video Disc or Digital Versatile Disc) drive may be connected instead of CD-ROM drive 42. The external storage of HDD40 or CD-ROM drive 42 grade is stored in the receipt place called the "media bay" of for example, 12 PCs inside of the body, or "device bay." The external storage these-equipped standardly may be attached possible [other equipments and exchange like FDD or a battery pack], and exclusively.

[0047] In addition, main memory 16 corresponds to the main storage concerning this invention, and HDD40 corresponds to the secondary storage concerning this invention.

[0048] Moreover, the USB port is established in the multirole PCI device 38, and this USB port is connected with the USB connector 44 prepared in the wall surface of PC12 main part etc. USB is supporting the function (hot plugging function) of taking out and inserting a new peripheral device (USB device) with powering on, and the functional (plug and play) function to carry out automatic recognition of the newly connected peripheral

device, and to reconfigure a system configuration. Daisy chain connection of a maximum of 63 USB devices can be made to one USB port. A keyboard, a mouse, a joy stick, a scanner, a printer, a modem, a display monitor, the tablet of the example of a USB device, etc. are various.

[0049] Furthermore, EEPROM50 is connected to the multirole PCI device 38 through SM bus. EEPROM50 is the memory for holding information, such as a password registered by the user, a supervisor password, and a product serial number, it is nonvolatile and rewriting of the contents of storage is enabled electrically.

[0050] I/O bus 22 is a bus where a data transfer rate is lower than PCI bus 20 (for example, bus width of face of 16 bits, maximum-data-rate 4MBps), and is Super. In addition to the flash ROM 56 and CMOS58 which consist of I/O controller 46, the power supply controller 48, EEPROM, etc., it is used for connecting a real-time clock (RTC) and the peripheral devices (all being illustration abbreviations) like a keyboard / mouse controller which operate comparatively at a low speed.

[0051] Super I/O Port 52 is connected to I/O controller 46, and it is a circumference controller for controlling the drive of a floppy disk drive (FDD), I/O (PIO) of the parallel data through the parallel port, and I/O (SIO) of the serial data through the serial port.

[0052] The power supply controller 48 can mainly perform the power management of computer system 10, and thermal management, and can constitute them from a single chip microcomputer equipped with MPU, RAM, ROM, the timer, etc. The program required to perform a power management and thermal management and the reference table are stored in ROM. The power supply controller 54 is connected to the power supply controller 48. The DC to DC converter for generating the direct-current constant voltage of 5V and 3.3V grade used for the power supply controller 54 by the battery charger for charging a battery and computer system 10 is contained, and power control is performed under the power supply controller 48.

[0053] A flash ROM 56 is the memory for holding the program of firmwares, such as BIOS and a bootstrap code, it is nonvolatile and rewriting of the contents of storage is enabled electrically. Moreover, it connects with a backup power supply, volatile semiconductor memory is constituted, and CMOS58 functions as a high-speed nonvolatile and storage means.

[0054] In addition, in order to constitute computer system 10, many electrical circuits are required also besides having been shown in drawing 1 . However, these are common knowledge, and since they do not constitute the summary of this invention to this contractor, they omit explanation in this specification to him. Moreover, in order to avoid complication of a drawing, it writes not illustrating a part of connection during each hardware block in drawing, either in addition.

[0055] Next, boot of the 1st OS first performed by turning on the electric power switch of computer system 10 as an operation of this operation form is explained with reference to the flow chart of drawing 3 . In addition, as the 1st OS, OS equipped with a file system like "OS/2" of "Windows 98 or NT", and U.S. IBM of U.S. Microsoft Corp., for example is applicable.

[0056] If an electric power switch is turned on, the field where the program of POST (Power On Self Test) which is a part of BIOS among the storage regions of a flash ROM 56 is memorized will be accessed, and the program of POST will be performed. As shown in drawing 3 as a "system test / initialization code", while each hardware of computer system 10 is tested by this, initialization (clearance of the contents of storage) of main memory 16 is performed (Step 100), and initialization (specifically initialization of an external hardware interrupt vector, initialization of external hardware, initialization of a software interrupt vector, etc.) of the hardware environment of computer system 10 is performed continuously (Step 102).

[0057] Moreover, the field (refer to drawing 9 as [System BIOS field :] an example) for memorizing firmwares, such as BIOS containing POST etc., a bootstrap code, and secondary-storage I/O service, is established in main memory 16. After POST copies these firmwares to the system BIOS field of main memory 16 from a flash ROM 56, it is jumped to the field where the bootstrap code is memorized among the system BIOS fields on main memory 16 as it is describing "It jumps to a bootstrap code" at drawing 3 .

[0058] According to the priority which the bootstrap code was performed and was defined beforehand first by this, two or more boot drives (A drive [Drive of the secondary storage in which the boot image of OS may be

stored (FDD and HDD40 grade) : usually] (FDD) top priority) are accessed in order, and it is searched for the boot image of OS which should be booted. With this operation form, the boot image of the 1st OS is memorized by HDD40, and if FD is not set to FDD, the boot image of the 1st OS memorized by HDD40 is discovered as a boot image of OS which should be booted. and a series of instruction codes (loader [of the 1st OS]: -- located in the head of the boot image of the 1st OS) for loading the 1st OS to main memory 16 among the boot images of the 1st OS are loaded to main memory 16 from HDD40 (Step 106)

[0059] After the processing in a bootstrap code is completed, it jumps at the head of the field which loaded the loader of the 1st OS among the storage regions of main memory 16, and the program (a series of instruction codes) of the loader of the 1st OS is performed as it is describing "It jumps to the loader of the 1st OS" at drawing 3 . Thereby, a series of instruction codes which are equivalent to 1st OS main part among the boot images of the 1st OS are loaded to main memory 16 from HDD40 (Step 110).

[0060] After processing by the loader of the 1st OS is completed, among the storage regions of main memory 16, by the loader and OS, it jumps to the start address of the 1st OS fixed beforehand, and the program (a series of instruction codes) of the 1st OS is performed as it is describing "It jumps to the start of the 1st OS" at drawing 3 . The 1st OS is booted by this and it will be in the state where the 1st OS is working on computer system 10.

[0061] Next, the 1st OS and the 2nd different OS are made to boot from the state where the 1st OS is working, and a series of processings in which predetermined application is performed under control of the 2nd OS are explained with reference to the flow chart of drawing 4 </A> (and drawing 5). In addition, below, the case where DOS is applied is explained as an example of the 2nd OS.

[0062] A series of processings shown in drawing 4 (and drawing 5) start execution, when a system or a user starts OS change bootstrap. In addition, OS change bootstrap is acquirable by downloading from a server like Step 130 mentioned later. Moreover, OS change bootstrap may be beforehand installed in the HDD40 grade as an application program performed under control of the 1st OS.

[0063] At Step 130, the information for booting the 2nd OS and performing predetermined application under control of the 2nd OS is downloaded from a server via a network (for example, Internet) under control of the 1st OS. In addition, in this operation form, the information for booting the 2nd OS consists of a boot image file of the 2nd OS, a device driver, and a secondary-storage I/O patch program, and the predetermined application program which should be performed under control of the loader of the 2nd OS, 2nd OS main part, and 2nd OS is contained in the boot image file of the 2nd OS.

[0064] There are various methods as the method of download, for example, the user of computer system 10 starts a browser (browser) etc., and there is the method of downloading by perusing the homepage for performing download offered by the server. In the homepage shown in drawing 6 as an example, the name of two or more sorts of application programs (three kinds of programs for performing the program for performing program for performing a benchmark test to computer system 10 in the example of drawing 6 and renewal of BIOS and the self-test of computer system 10) which can be downloaded by the execute permission under control of the 2nd OS is displayed respectively.

[0065] In this case, if the name of an application program with which the user is asking for execution is chosen by the user from the names of two or more sorts of application programs as an example OS change bootstrap downloads from a server as a binary file to computer system 10 first. By starting OS change bootstrap on computer system 10, and performing Step 130 The boot image file of the 2nd OS containing the selected application program, a device driver, and a secondary-storage I/O patch program can download as a binary file in order.

[0066] Moreover, as the method of other downloads, as shown in drawing 7 as an example, there are a boot image file of the 2nd OS, a device driver, and a method of receiving a secondary-storage I/O patch program (and OS change bootstrap) as an attached file of an E-mail.

[0067] In the example of drawing 7 , by the transmitting person, OS change bootstrap (in drawing 7 , it is written as "doboot.exe"), a device driver (drawing 7 -- "dosboot.sys" and "DOSBOOT.vxd", and the notation) -- A

secondary-storage I/O patch program (in drawing 7 , it is written as "dosboot.com"), And the E-mail to which the boot image file (it is written as "biosupdt.img" in drawing 7) of the 2nd OS containing the application program which updates BIOS was respectively appended as a binary file, and was transmitted is received by computer system 10. The screen image in the state where the E-mail which received was opened is shown. These binary files are once downloaded to the temporary directory of a secondary storage, if the icon of OS change bootstrap "doboot.exe [bootstrap]" was written is clicked, OS change bootstrap will be loaded to main memory 16, and processing after the following step 132 will be performed.

[0068] In addition, each of various kinds of downloads mentioned above is downloads performed under control of the 1st OS, and they can be downloaded by the general-purpose protocol, without being dependent on the special protocol for making it boot by RIMOTO. Moreover, by the 1st OS, the device driver and secondary-storage I/O patch program which were downloaded, and the boot image file of 2nd OS are recognized as a file, and are once memorized by HDD40 under management of the file system of the 1st OS.

[0071] If OS change bootstrap starts a device driver, it will require the hook of OS end from a device driver at the following step 138. Thereby, a device driver performs processing which hooks the end of the 1st OS. Namely, if the reboot of computer system 10 and execution of the end processing for power supply OFF are directed, the 1st OS Terminate all the application programs started and all the files currently opened are closed. A series of OS end processings in which data are saved if needed are performed, after OS end processing is completed, it jumps to the program (this is also a part of OS) which performs a reboot or power supply OFF of computer system, and a reboot or power supply OFF of computer system is performed.

[0072] The jump to the program which reboots computer system (or power supply OFF) is an indirect jump jumped at the jump place registered into the table with reference to the table on which the jump place was registered. Instead of control of CPU14 jumping to the program which reboots computer system (or power supply OFF), after hooking the end of the 1st OS and completing OS end processing by rewriting the jump place registered into the aforementioned table, a device driver is jumped to the field to which the device driver is loaded among the storage regions of main memory 16, and it is made to move from it to oneself (device driver).

[0073] Moreover, a device driver will be set to the following step 148, if processing which hooks the end of the 1st OS is performed. It requests to secure a work field to the storage region of main memory 16 to the 1st OS. The secondary-storage I/O patch program memorized by HDD40 as a file, The boot image file (binary file to which all change from a series of instruction codes) of the 2nd OS It loads to the work field (field which is not overwritten under control of the 1st OS) secured by the 1st OS (see a "secondary-storage I/O patch program" and a "DOS boot image" of drawing 9 (A)). And a device driver interrupts processing, after performing Step 148.

[0074] On the other hand, OS change bootstrap will require OS end from the 1st OS in the following step 140, if the hook of OS end is required from a device driver. [whether this OS end demand calls the service of OS whose OS change bootstrap itself directs the end of the 1st OS, and] Or the end of the 1st OS is directed by interactive operation of a user (for example, if the end of OS under "present operation is directed to a user). The guidance message of downloaded application being performed" is displayed on LCD28, and it is carried out by the thing -- according to this guidance message, a user directs the end of OS. If the end of OS is required as mentioned above, OS change bootstrap will end processing.

[0075] Although the usual processing (Step 156) is performed in between [until there is an OS end demand (the judgment of Step 158 is affirmed)], the 1st OS under operation If OS end is directed from OS change bootstrap, the judgment of Step 158 will be affirmed. Like the case where a reboot and power supply OFF of computer system 10 are directed in the usual procedure All the application programs started are terminated, all the files currently opened are closed, and a series of OS end processings in which data are saved if needed are performed (Step 160).

[0076] when a reboot and power supply OFF of computer system 10 are directed in the usual procedure here, the hook (Step 146) of OS end is performed -- ****'s (the judgment of Step 162 denies) -- it jumps to the program which performs a reboot or power supply OFF of computer system, and a reboot or power supply OFF

of computer system is performed However, when the hook of OS end is performed (the judgment of Step 162 is affirmed), if OS end processing is completed, processing of a device driver will be resumed because control of CPU14 moves to a device driver.

[0077] By the way, DOS used as the 2nd OS in this operation gestalt As opposed to the mode of a processor (CPU14) operating in the state of the real mode Other OS's ("Windows 98 or NT" and "OS/2" which were mentioned as an example of the 1st OS in this operation gestalt, or other OS's) applicable as the 1st OS operate in the state where the mode of a processor is the modes other than the real mode. For this reason, it is necessary to change the mode of a processor in booting the 2nd OS. However, when the mode of a processor is changed, the virtual-memory mechanism of a processor may become effective, or may become invalid (as an example, when the mode of a processor is changed into the real mode from the mode in which OS's, such as "Windows", can operate, a virtual-memory mechanism changes to an effective -> invalid).

[0078] Therefore, the rebooted device driver In the following step 150, as pretreatment for changing the mode of a processor A series of program codes (a series of instruction codes which make the mode of a processor change) for changing the mode of a processor into the mode (this operation gestalt real mode) in which the 2nd OS can operate The virtual address and a physical address load to the same field (a virtual-memory mechanism is effective -> field recognized as the same field even if it changes to an invalid (or the reverse)) among the storage regions of main memory 16.

[0079] Moreover, it needs to be accessible in the mode in which the 2nd OS operates, also about the secondary-storage I/O patch program already loaded to main memory 16, and the boot image file of 2nd OS. For this reason, at Step 150, the field which is continuing physically on the storage region of main memory 16 is secured, and a secondary-storage I/O patch program and the boot image file of 2nd OS are transmitted to the field which carried out [aforementioned] reservation (relocation).

[0080] If the above-mentioned processing is performed, it will jump to the field which loaded a series of program codes at Step 150 among the storage regions of main memory 16, and a series of aforementioned program codes will be performed. Thereby, the own mode of a processor changes to the mode (real mode) in which the 2nd OS can operate (Step 152). In addition, since the virtual address and the physical address are loaded to the same field, a series of program codes which make the mode of a processor change can perform a series of program codes normally to the last irrespective of changes in the mode of a processor.

[0081] If a series of program codes are performed to the last and change in the mode of a processor is completed, it will jump to the field to which the secondary-storage I/O patch program was transmitted at previous Step 150 among the storage regions of main memory 16, and a secondary-storage I/O patch program will be started as it is describing "It jumps to a patch program" at drawing 4 . Hereafter, the processing realized by the secondary-storage I/O patch program is explained with reference to the flow chart of drawing 5 .

[0082] At Step 170, instead of POST which is a part of BIOS, hardware environment is initialized as if the firmware of computer system 10 carried out the LDA of the 2nd OS (specifically initialization of an external hardware interrupt vector, initialization of external hardware, initialization of a software interrupt vector, etc.). (as if the 2nd OS was loaded as the 1st OS) In addition, about the secondary-storage I/O-service interrupt vector table which is a part of software interrupt vector, reinitialization is carried out so that it may mention later.

[0083] Moreover, in order to boot the 2nd OS, the boot image file of the 2nd OS and secondary-storage I/O patch program which are memorized by main memory 16 need to be made not to be destroyed by loading of 2nd OS main part to main memory 16 (overwrite). For this reason, at the following step 172, the boot image file of the 2nd OS and the secondary-storage I/O patch program itself are transmitted to the field (field which 2nd OS main part is not loaded and is not used as a work field of the 2nd OS) which is not overwritten by loading of the 2nd OS.

[0088] In addition, about the boot image file of the 2nd OS, it transmits to the field of the high order address which was far apart from the head of main memory 16 4 M bytes or more, for example. It can also prevent this

field that the boot image file of the 2nd OS is overwritten and destroyed by the hook (it mentions later for details) of a loading memory size query software (memory size query software) interrupt handler since a DOS main part is not loaded or it is not used as a work field of DOS.

[0089] Moreover, in the field of the predetermined size which makes a head address 0h of main memory 16, the table (interrupt vector table) which the jump place when a software interrupt etc. occurs interrupted, and was memorized for every factor is memorized. An interrupt vector table is initialized so that it may jump to the firmware (secondary-storage I/O service (INT13h handler)) memorized to the system BIOS field, when secondary-storage I/O-service interruption (interruption of interruption factor =INT13h) occurs among various kinds of software interrupts, and the secondary storage specified via the secondary-storage I/O service of a firmware is accessed.

[0090] On the other hand, at the following step 174, when secondary-storage I/O-service interruption occurs, a secondary-storage I/O-service software interrupt handler is hooked by rewriting an interrupt vector table so that it may jump to the INT13h handler which is a part of secondary-storage I/O patch program.

[0091] If the INT13h handler of a secondary-storage I/O patch program is started in jumping from an interrupt vector table, it will check the value (the value of DL register expresses the drive number of the secondary storage to access) of DL register inside a processor first. And the value of DL register controls to access the field where the boot image file of the 2nd OS is memorized among the storage regions of main memory 16, and to be read to be shown in drawing 10 at the time of the value (DL=0) showing the read-out service from the drive (usually FDD) of a secondary storage with the highest priority.

[0092] On the other hand, when the values of DL register are other values (DL!=0), it controls to jump to the secondary-storage I/O service of a firmware. In this case, as shown in drawing 10 , the drive of the secondary storage specified by DL register will be accessed.

[0093] Moreover, generally, computer system will offer the service (query of a loading memory size) which notifies the size of loading memory, if the software interrupt (interruption of interruption factor =INT15h) which asks the size of loading memory (main memory 16) is generated by the program executed by OS or OS. A loading memory size query software interrupt is generated at the time of initialization, a loading memory size is acquired, and there are some programs which are performed by OS or OS and which require that a field should be used as an own work field to the high limit address of loading memory.

[0094] For this reason, at the following step 176, when a loading memory size query software interrupt occurs, a loading memory size query software interrupt handler is hooked by rewriting an interrupt vector table so that it may jump to the INT15h handler which is a part of secondary-storage I/O patch program.

[0095] If the INT15h handler of a secondary-storage I/O patch program is started in jumping from an interrupt vector table, it will be changed into the value meaning the storage region of loading memory existing the loading memory size notified to interruption generating origin rather than the field where the boot image file of the 2nd OS is memorized only to the address (for example, address of the field separated from the head of main memory 16 4 M bytes) of lower order. Thereby, from interruption generating origin (program executed by OS or OS), it will seem that the field where the boot image file of the 2nd OS is memorized is not mounted, and it is prevented that the boot image file of the 2nd OS is overwritten.

[0096] It jumps to the field where the bootstrap code is memorized among the system BIOS storage regions of main memory 16, and a bootstrap code is performed as processing by the secondary-storage I/O patch program is completed by the above and it is describing "It jumps to a bootstrap code" at drawing 5 .

[0097] Although the value of DL register is set to 0 and secondary-storage I/O-service interruption (interruption of INT13h) is generated in order to check whether the boot image of OS which priority should boot to the highest boot drive thereby first exists Since secondary-storage I/O-service interruption is hooked by rewriting of an interrupt vector table as explained previously It goes via the INT13h handler of a secondary-storage I/O patch program. The field where the boot image file of the 2nd OS is memorized among the storage regions of main memory 16 is accessed, and the boot image of the 2nd OS is discovered as a boot image of OS which should be booted.

[0098] and a series of instruction codes (loader [of the 2nd OS]: — located in the head of the boot image of the 2nd OS) for loading 2nd OS main part to main memory 16 among the boot images of the 2nd OS are loaded to the predetermined field on main memory 16 (Step 180)

[0099] moreover, the head address of the boot image file of the 2nd OS -- for a secondary-storage I/O patch program -- obvious -- it is -- since -- a secondary-storage I/O patch program may access the field of the boot image file of the 2nd OS on the direct main memory 16, and loading (Step 180) of the loader of the 2nd OS which a bootstrap code performs may be realized by transmitting to the predetermined field of main memory 16 If it does in this way, the 2nd OS can be booted irrespective of the priority of a bootstrap code.

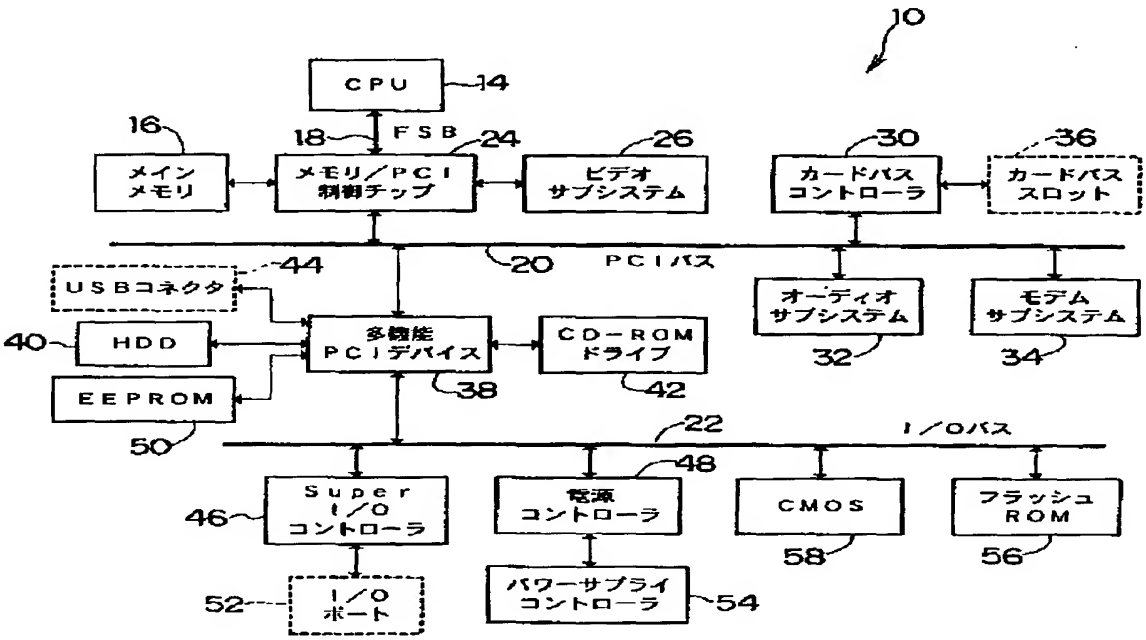
[0100] After the processing in a bootstrap code is completed, it jumps at the head of the field which loaded the loader of the 2nd OS among the storage regions of main memory 16, and the program (a series of instruction codes) of the loader of the 2nd OS is performed as it is describing "It jumps to the loader of the 2nd OS" at drawing 5 . Thereby, a series of instruction codes which are equivalent to 2nd OS main part among the boot images of the 2nd OS are loaded to the predetermined field on main memory 16 (field "the DOS main part and the work area [field]" are written to drawing 9 and drawing 10) (Step 182).

[0101] After processing by the loader of the 2nd OS is completed, among the storage regions of main memory 16, by the loader and OS, it jumps to the start address of the 2nd OS fixed beforehand, and the program (a series of instruction codes) of the 2nd OS is performed as it is describing "It jumps to the start of the 2nd OS" at drawing 5 . The 2nd OS is booted by this and it will be in the state where the 2nd OS is working on computer system 10.

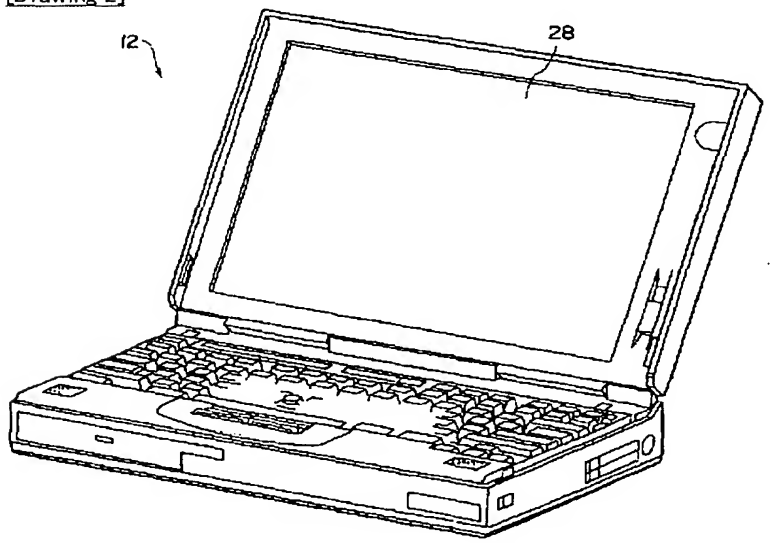
[0102] Moreover, the predetermined application program which should be performed under control of the 2nd OS is contained in the boot image file of the 2nd OS, if the 2nd OS is booted, a predetermined application program will be loaded to the predetermined field of main memory 16, and will be performed, and predetermined processings (for example, renewal of the benchmark test and BIOS to computer system 10, the self-test of computer system 10, etc.) will be performed (Step 186). In addition, it may be made to perform loading and execution of this application program automatically, and may be made to perform them according to directions by the user.

DRAWINGS

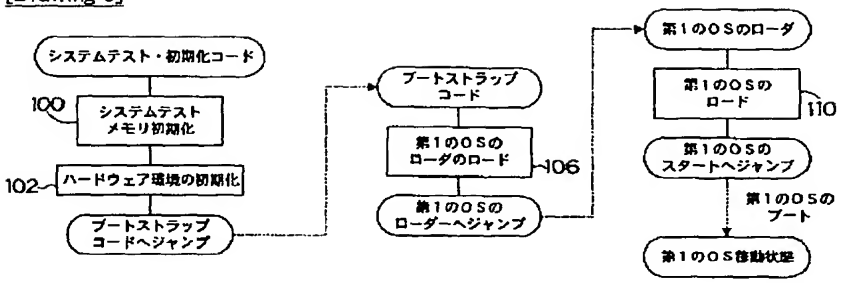
[Drawing 1]



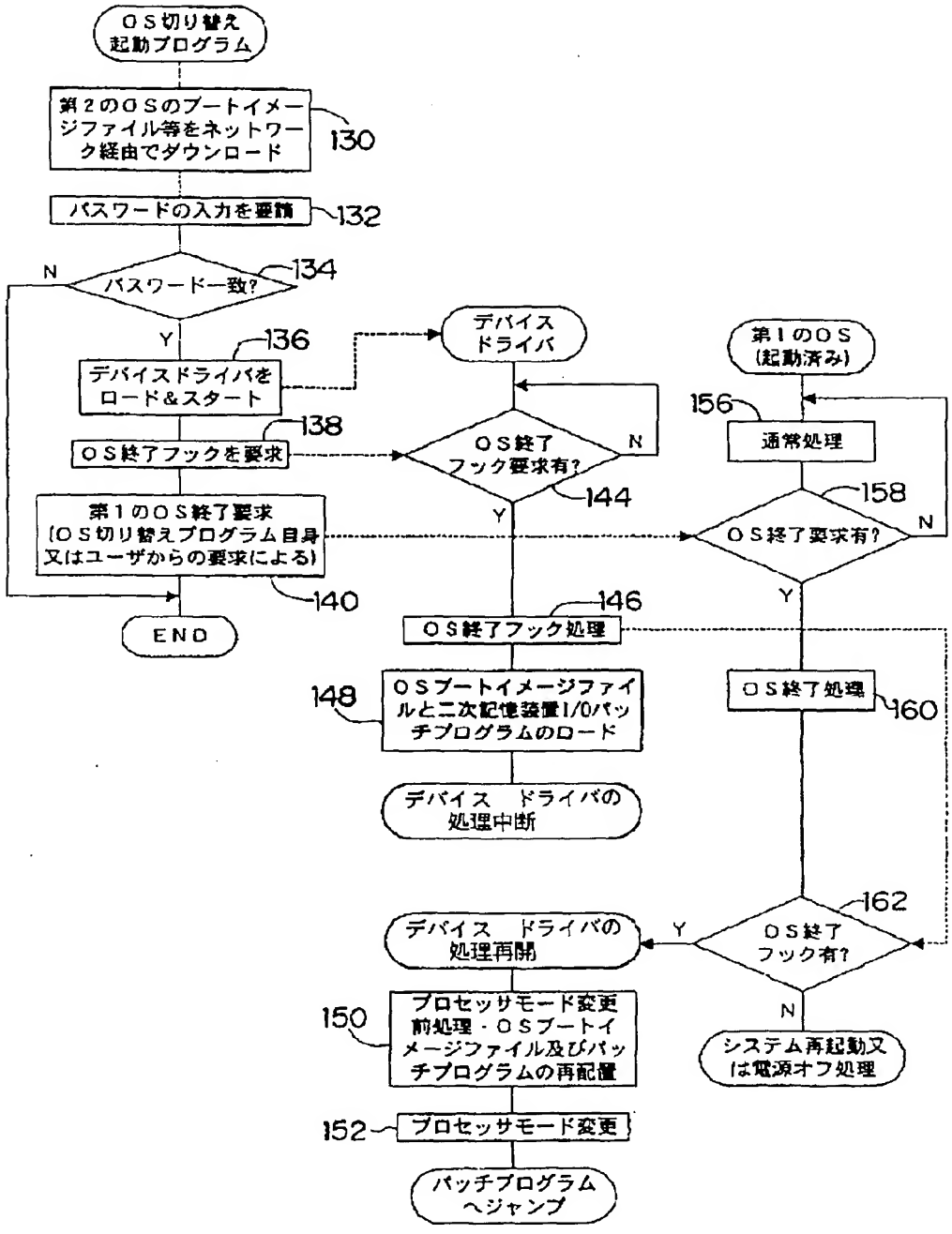
[Drawing 2]



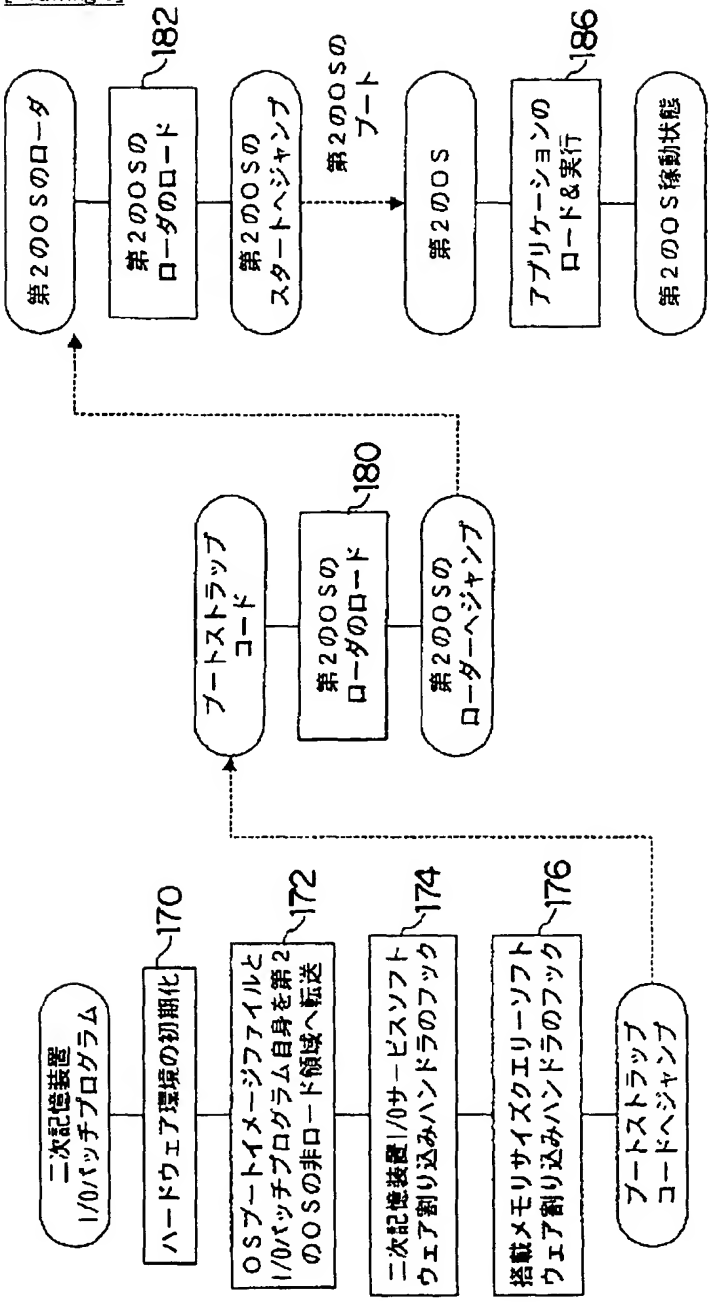
[Drawing 3]


システムテスト・初期化コード
100
システムテスト
メモリ初期化
102
ハードウェア環境の初期化
ブートストラップ
コードへジャンプ
ブートストラップ
コード
第1のOSの
ローダのロード
106
第1のOSの
ローダーへジャンプ
第1のOSのローダ
第1のOSの
ロード
110
第1のOSの
スタートへジャンプ
第1のOSの
ブート
第1のOS稼動状態

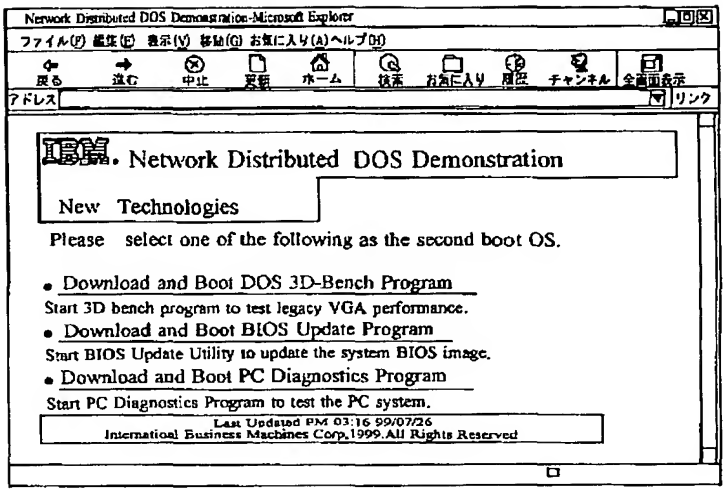
[Drawing 4]


ＯＳ切り替え
起動プログラム
第２のＯＳのブートイメージファイル等をネットワーク経由でダウンロード
130
パスワードの入力を要請
132
N
パスワード一致?
134
Y
136
デバイスドライバをロード＆スタート
デバイス
ドライバ
138
ＯＳ終了フックを要求
ＯＳ終了
フック要求有?
N
144
Y
第１のＯＳ終了要求
(ＯＳ切り替えプログラム自身又はユーザからの要求による)
140
END
第１のＯＳ
(起動済み)
156
通常処理
158
ＯＳ終了要求有?
N
Y
146
ＯＳ終了フック処理
148
ＯＳブートイメージファイルと二次記憶装置I/Oパッチプログラムのロード
デバイス　ドライバの
処理中断
ＯＳ終了処理
160
162
デバイス　ドライバの
処理再開
Y
ＯＳ終了
フック有?
N
150
プロセッサモード変更前処理・ＯＳブートイメージファイル及びパッチプログラムの再配置
システム再起動又は電源オフ処理
152
プロセッサモード変更
パッチプログラム
へジャンプ

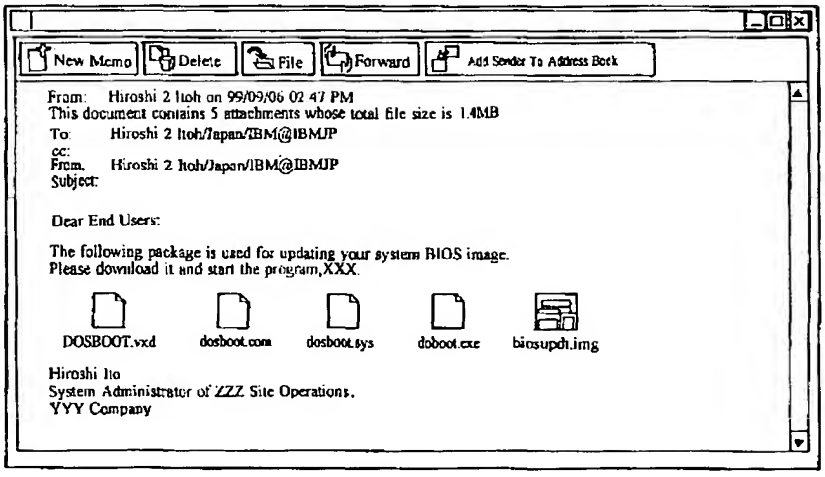
[Drawing 5]


二次記憶装置
I/Oパッチプログラム
ハードウェア環境の初期化
170
OSブートイメージファイルと
I/Oパッチプログラム自身を第2
のOSの非ロード領域へ転送
172
二次記憶装置I/Oサービスソフト
ウェア割り込みハンドラのフック
174
搭載メモリサイズクエリーソフト
ウェア割り込みハンドラのフック
176
ブートストラップ
コードへジャンプ
ブートストラップ
コード
第2のOSの
ローダのロード
180
第2のOSの
ローダーへジャンプ
第2のOSのローダ
第2のOSの
ローダのロード
182
第2のOSの
スタートへジャンプ
第2のOSの
ブート
第2のOS
アプリケーションの
ロード&実行
186
第2のOS稼動状態

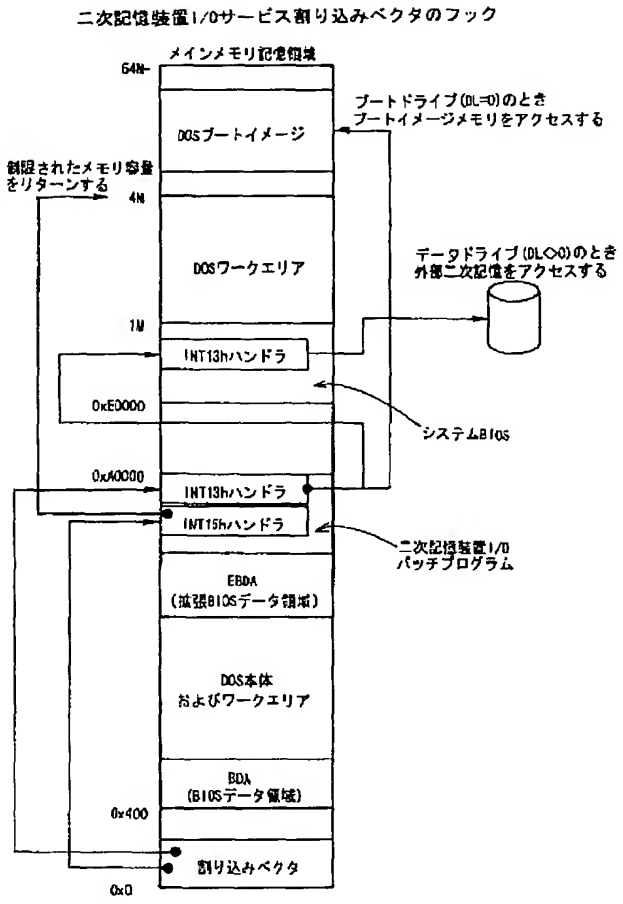
[Drawing 6]



[Drawing 7]



[Drawing 10]


二次記憶装置I/Oサービス割り込みベクタのフック
メインメモリ記憶領域
64M
ブートドライブ(DL=0)のとき
ブートイメージメモリをアクセスする
DOSブートイメージ
制限されたメモリ容量
をリターンする
4M
DOSワークエリア
データドライブ(DL<>0)のとき
外部二次記憶をアクセスする
1M
INT13hハンドラ
0xE0000
システムBIOS
0xA0000
INT13hハンドラ
INT15hハンドラ
二次記憶装置I/O
パッチプログラム
EBDA
(拡張BIOSデータ領域)
DOS本体
およびワークエリア
BDA
(BIOSデータ領域)
0x400
割り込みベクタ
0x0

(19)日本国特許庁（ＪＰ）　(12) **公開特許公報**（Ａ）　(11)特許出願公開番号
特開2001－100983
（P2001－100983A）
(43)公開日　平成13年４月13日(2001.4.13)

(51)Int.Cl.⁷		識別記号	FI			ﾃｰﾏｺｰﾄﾞ(参考)
G06F	9/06	410	G06F	9/06	410D	5B076
	9/46	340		9/46	340A	5B098

審査請求　有　　請求項の数14　ＯＬ　（全 20 頁）

(21)出願番号　特願平11－273860

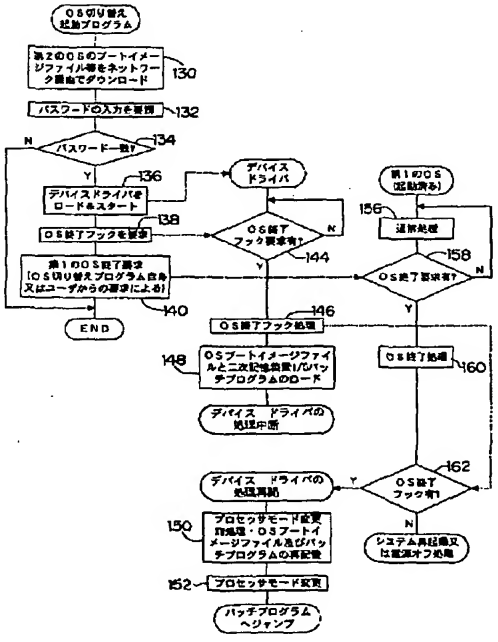
(22)出願日　平成11年９月28日(1999.9.28)

(71)出願人　390009531
インターナショナル・ビジネス・マシーンズ・コーポレーション
INTERNATIONAL BUSINESS MASCHINES CORPORATION
アメリカ合衆国10504、ニューヨーク州　アーモンク　（番地なし）
(72)発明者　伊藤　浩
神奈川県大和市下鶴間1623番地14　日本アイ・ビー・エム株式会社　大和事業所内
(74)代理人　100086243
弁理士　坂口　博　（外５名）

最終頁に続く

(54)【発明の名称】　コンピュータの制御方法、コンピュータ及び記録媒体

(57)【要約】
【課題】　コンピュータ上でブートさせるＯＳの切り替えを容易に実現できる汎用的な方法を提供する。
【解決手段】　コンピュータ上で第１のＯＳが稼動している状態で、第２のＯＳをブートさせるための情報を、ホームページの閲覧や電子メールの受信等によってダウンロードし(130)、パスワード照合後にデバイスドライバを起動し、ＯＳ終了のフックを要求する(134～138)。デバイスドライバは、要求に応じてＯＳ終了をフックする処理を行い、ダウンロードしたブートイメージファイル等をメインメモリにロードした後に処理を中断する(144～148)。ＯＳが終了するとデバイスドライバの処理が再開され、プロセッサのモードを変更(150,152)した後にバッチプログラムを起動する。これにより、各種の前処理が行われた後に、メインメモリにロードしたブートイメージファイルにより第２のＯＳがブートされる。



1

【特許請求の範囲】

【請求項１】　コンピュータ上で第１のＯＳが稼動している状態で、前記第１のＯＳと異なる第２のＯＳをブートさせるための情報を第１のＯＳの制御下で取得し、
前記取得した情報を前記コンピュータの主記憶装置に書き込み、
第１のＯＳの終了を検知すると、前記主記憶装置に書き込んだ情報を消去することなく、前記主記憶装置に書き込んだ情報により前記コンピュータ上で第２のＯＳをブートさせるコンピュータの制御方法。

【請求項２】　前記第２のＯＳをブートさせるための情報を、前記第１のＯＳの制御下で、通信回線を介して前記コンピュータの外部よりファイルとして受信することにより取得することを特徴とする請求項１記載のコンピュータの制御方法。

【請求項３】　前記取得した情報を前記コンピュータの二次記憶装置に一旦記憶することを特徴とする請求項２記載のコンピュータの制御方法。

【請求項４】　前記第２のＯＳをブートさせるための情報は前記コンピュータの二次記憶装置に事前に書き込まれており、
前記第２のＯＳをブートさせるための情報を、前記第１のＯＳの制御下で、前記二次記憶装置から読み出すことにより取得することを特徴とする請求項１記載のコンピュータの制御方法。

【請求項５】　前記第２のＯＳをブートさせるための情報には、前記第２のＯＳのブート・イメージ・ファイルが含まれており、
前記第２のＯＳのブート・イメージ・ファイルを前記コンピュータの主記憶装置に書き込み、前記第１のＯＳの終了を検知した後に、ファームウェアのブート・ストラップ・コードによって前記主記憶装置の前記第２のＯＳのブート・イメージ・ファイルを書き込んだ領域がアクセスされるようにすることで、前記コンピュータ上で前記第２のＯＳをブートさせることを特徴とする請求項１記載のコンピュータの制御方法。

【請求項６】　前記第２のＯＳのブート・イメージ・ファイルが、二次記憶装置用に作成されたものであることを特徴とする請求項５記載のコンピュータの制御方法。

【請求項７】　ファームウェアのブート・ストラップ・コードによって前記主記憶装置の前記第２のＯＳのブート・イメージ・ファイルを書き込んだ領域がアクセスされるようにすることを、二次記憶装置Ｉ／Ｏサービス・コードをフックすることにより行うことを特徴とする請求項５記載のコンピュータの制御方法。

【請求項８】　前記第２のＯＳをブートさせるための情報を、前記主記憶装置の記憶領域のうち第１のＯＳの制御下で上書きされない領域に一旦書き込んだ後に、前記書き込んだ情報の一部又は全部を、前記主記憶装置の記憶領域のうち少なくとも第２のＯＳのブートによって上

2

書きされない領域に再配置することを特徴とする請求項１記載のコンピュータの制御方法。

【請求項９】　前記コンピュータ上で第２のＯＳをブートさせた後に、第２のＯＳが稼動している状態で、ファームウェアを更新するプログラム、特別なプロトコルを必要とするネットワーク・ブートを行うプログラム、セットアップを行うプログラム、コンピュータ・システムの診断を行うプログラム、第１のＯＳの稼動状態では動作が保証されない、或いは実行に適さない処理を行うプログラム、の少なくとも１つを実行させることを特徴とする請求項１記載のコンピュータの制御方法。

【請求項１０】　コンピュータ上での第１のＯＳのブートから、少なくとも前記コンピュータ上での第２のＯＳのブートに至る一連の処理を自動的に連続して行なわせることを特徴とする請求項１記載のコンピュータの制御方法。

【請求項１１】　前記第１のＯＳと異なる第２のＯＳをブートさせるための情報として、互いに異なる複数種のＯＳをブートさせるための情報を各々取得し、
ＯＳの終了を検知する毎に、前記複数種のＯＳのうちブート未実行のＯＳを前記コンピュータ上でブートさせることを特徴とする請求項１記載のコンピュータの制御方法。

【請求項１２】　第２のＯＳをブートさせるための情報を取得する前、又は後に、第１のＯＳ上で、コンピュータの利用者が第２のＯＳの正当な利用者か否か、又は第２のＯＳが正当なＯＳであるか否かを確認することを特徴とする請求項１記載のコンピュータの制御方法。

【請求項１３】　第１のＯＳが稼動している状態で、前記第１のＯＳと異なる第２のＯＳをブートさせるための情報を第１のＯＳの制御下で取得する取得手段と、
前記取得手段によって取得された情報を前記コンピュータの主記憶装置に書き込む書込手段と、
第１のＯＳの終了を検知すると、前記書込手段によって主記憶装置に書き込まれた情報を消去することなく、前記主記憶装置に書き込まれた情報により前記コンピュータ上で第２のＯＳをブートさせるブート制御手段と、
を含むコンピュータ。

【請求項１４】　コンピュータ上で第１のＯＳが稼動している状態で、前記第１のＯＳと異なる第２のＯＳをブートさせるための情報を第１のＯＳの制御下で取得する第１のステップ、
前記取得した情報を前記コンピュータの主記憶装置に書き込む第２のステップ、
第１のＯＳの終了を検知すると、前記主記憶装置に書き込んだ情報を消去することなく、前記主記憶装置に書き込んだ情報により前記コンピュータ上で第２のＯＳをブートさせる第３のステップを含む処理をコンピュータに実行させるためのプログラムが記録された記録媒体。

【発明の詳細な説明】

【０００１】
【発明の属する技術分野】本発明はコンピュータの制御方法、コンピュータ及び記録媒体に係り、特に、コンピュータ上で通常ブートされる第１のＯＳと異なる第２のＯＳをブートさせるコンピュータの制御方法、前記制御方法が適用されたコンピュータ、及びコンピュータによって前記制御方法を実現するためのプログラムが記録された記録媒体に関するに関する。
【０００２】
【従来の技術】コンピュータのオペレーティング・システム（Operating System、以下「ＯＳ」という）としては、現在、多種多様なＯＳが存在しており、それぞれのＯＳは互いに異なる特徴を有している。このため、起動時に特定のＯＳが固定的にブートされるように設定されたコンピュータであっても、他のＯＳをブートする必要が生ずることがある。例えばＢＩＯＳ等のファームウェアを更新（アップデート）するアプリケーション・プログラムは、複数のタスクを並列に実行可能な機能を備えたＯＳ（例えば米マイクロソフト社の"Ｗｉｎｄｏｗｓ"等）が稼動している環境下で実行するよりも、単一のタスクのみ実行可能なＯＳ（例えばＤＯＳ（Disk Operating System））が稼動している環境下で実行することが望ましい。
【０００３】コンピュータの起動時に、通常ブートされるＯＳ（以下、便宜的に第１のＯＳという）と異なるＯＳ（以下、便宜的に第２のＯＳという）をブートさせるための方法の１つとして、第２のＯＳをブートさせるためのブート・イメージ（先頭から順に読み出して実行することでＯＳがブートされる一連の命令コード）をフロッピー（登録商標）ディスク（ＦＤ）に記憶しておき、該ＦＤをフロッピーディスク・ドライブ（ＦＤＤ）にセットしてコンピュータ・システムを起動することで第２のＯＳをブートさせる方法が知られている。
【０００４】また、他の方法として、コンピュータに接続されているネットワーク・アダプタ内蔵のファームウェアが、ＯＳと無関係にネットワークを介してサーバと通信を行って第２のＯＳをブートさせるための情報をサーバから取得し、コンピュータ上で第２のＯＳをリモートでブートさせる技術も知られている。
【０００５】
【発明が解決しようとする課題】しかしながら、ＦＤをＦＤＤにセットする方法はＦＤＤが設けられていないコンピュータには適用できないという欠点がある。また、第２のＯＳを起動するためのＦＤを作成したり、作成したＦＤをＦＤＤにセットしたり等の作業も煩雑であり、作業対象のコンピュータの数が多くなるに従って作業の煩雑さは非常に顕著となる。
【０００６】また、ネットワーク・アダプタ内蔵のファームウェアがサーバと通信を行ってサーバから情報を取得する方法では、ファームウェアとサーバとの通信が、ブートを行うことを前提とした特別なブート・ストラップ・プロトコル（例えばＩＥＴＦ(Internet Engineering Task Force)が規定しているＲＦＣ９５１(InternetRequest For Comments 951)におけるブート・ストラップ・プロトコル等）によって実現されるので、サーバが前記特別なプロトコルをサポートしている必要があると共に、ファイヤウォールやルータ等のネットワーク構成要素の影響により、ブートさせるための情報の取得に制限があり、汎用性に乏しいという問題がある。
【０００７】また、第２のＯＳをブートさせる他の方法として、ハードディスク等の記憶領域を複数のパーティションに分割し、各パーティションに、互いに異なるＯＳをブートさせるためのブート・イメージを記憶しておき、コンピュータの起動時に何れのパーティションにアクセスしてブートするかを定めたマスタ・ブート・レコードを書き替えた後にコンピュータを起動することによって、複数種のＯＳを選択的にブートさせる方法も考えられる。
【０００８】しかしながら、この方法では第２のＯＳをブートするためのブート・イメージがハードディスク等の記憶領域内に常駐することになるので、第２のＯＳの使用頻度が低い場合であっても、複数のパーティションのうちの１つが第２のＯＳによって占有されてしまい、ハードディスク等の記憶領域を有効に利用することができないという問題や、第１のＯＳがインストールされた後は第２のＯＳ用のパーティションをつくるのが容易ではないという問題がある。
【０００９】また、ハードディスクに第２のＯＳのブート・イメージを記憶する方法では、第２のＯＳの制御下で実行すべきアプリケーション・プログラム等も併せて記憶することが一般的である（ＦＤに記憶させる方法も同様）が、例えばＯＳを切り替えて実行すべき処理の一例であるファームウェアの更新は、ファームウェアのバージョン・アップ等がある毎に、バージョン・アップ後の最新のファームウェアへ更新するための最新のアプリケーション・プログラムを入手して行う必要があるので、このようなアプリケーション・プログラムをハードディスクに常駐させておいてもメリットは殆どない。また、アプリケーション・プログラムを実行した後にハードディスクから消去することも考えられるが、消去する作業が煩雑であるという問題もある。また、一般的に第１のＯＳと第２のＯＳでファイル・システムが異なる場合には、第１のＯＳ上のアプリケーションから第２のＯＳパーティション上のアプリケーションの更新並びに消去が困難となる。
【００１０】本発明は上記事実を考慮して成されたもので、コンピュータ上でブートさせるＯＳを切り替えることを容易に実現でき、汎用性に優れたコンピュータの制御方法を得ることを目的とする。
【００１１】更に、既存のコンピュータにハードウェア

やファームウェアの変更を加えることなく、ブートさせるＯＳを切り替える方法を提供することを目的とする。

【００１２】また本発明は、ブートさせるＯＳを容易に切り替え可能なコンピュータ及び記録媒体を得ることを目的とする。

【００１３】

【課題を解決するための手段】本発明に係るコンピュータの制御方法は、コンピュータ上で第１のＯＳが稼動している状態で、第１のＯＳと異なる第２のＯＳをブートさせるための情報を第１のＯＳの制御下で取得する。このように、情報の取得を第１のＯＳの制御下で行うことにより、取得した情報は第１のＯＳによってファイルとして認識され、第１のＯＳのファイル・システムによって管理されることになる。

【００１４】なお、第２のＯＳをブートさせるための情報は、例えば第２のＯＳをブートさせるためのブート・イメージ・ファイル（ブート・イメージを表すデータから成るファイル）を含んで構成することができる。また、第２のＯＳをブートさせるための情報をコンピュータの主記憶装置に書き込むためのプログラム（このプログラムはコンピュータに固定的に記憶されていてもよい）や、第２のＯＳをブートした後に実行すべきプログラムも含まれていてもよい。

【００１５】また、第２のＯＳをブートさせるための情報は、具体的には、例えば第１のＯＳの制御下で、通信回線を介してコンピュータの外部よりファイルとして受信することにより取得することができる。この場合、サーバとの間の通信が第１のＯＳの制御下で行われるので、第２のＯＳをブートさせるための情報を、リモートでブートさせるための特別なプロトコルに依存することなく、第１のＯＳの制御下で利用可能な汎用的な通信形態により容易に取得することができる（例えばWebブラウザによりWebサーバ（ネットワーク・ドライブ）からファイルとしてダウンロードしたり、電子メールの添付ファイルとして受信する等）。

【００１６】なお、第２のＯＳをブートさせるための情報を通信回線を介して外部よりファイルとして受信することにより取得する場合、取得した情報はメモリ等から成るコンピュータの主記憶装置に直ちに記憶するようにしてもよいが、ハードディスク等から成るコンピュータの二次記憶装置に一旦記憶するようにすれば、前記取得した情報を後に主記憶装置に書き込む迄の期間に、主記憶装置の記憶領域が圧迫されることを回避することができる。

【００１７】また、第２のＯＳをブートさせるための情報がコンピュータの二次記憶装置に事前に書き込まれている場合には、第２のＯＳをブートさせるための情報を、第１のＯＳの制御下で、二次記憶装置から読み出すことにより取得するようにしてもよい。この態様において、第２のＯＳをブートさせるための情報を二次記憶装置に事前に書き込んでおくための前記情報の入手は、通信回線を介して外部より前記情報を受信することで入手してもよいし、ＦＤ等の情報記憶媒体のドライブがコンピュータに設けられている場合には、前記情報が書き込まれた情報記憶媒体から前記情報を読み出すことで入手することも可能である。

【００１８】また本発明では、取得した情報をコンピュータの主記憶装置に書き込み、第１のＯＳの終了を検知すると、前記主記憶装置に書き込んだ情報を消去することなく、前記主記憶装置に書き込んだ情報によりコンピュータ上で第２のＯＳをブートさせる。コンピュータを再起動させるような場合には主記憶装置に記憶されている情報が消去されるが、本発明は主記憶装置に書き込んだ情報が消去されることはないので、主記憶装置に書き込んだ情報により第２のＯＳをブートさせることができる。

【００１９】また、主記憶装置に書き込んだ情報は、次にコンピュータを再起動したときには消去されるので、第２のＯＳをブートさせるために、例えば二次記憶装置の記憶領域を複数のパーティションに分割し、特定のパーティションに固定的に情報を記憶することで二次記憶装置の記憶領域が圧迫されたり、マスタ・ブート・レコードを書き替える、或いはＦＤをセットしてコンピュータを起動する等の煩雑な作業を行う必要もない。また、コンピュータにＦＤＤが設けられていない場合にも実施可能である。

【００２０】従って、本発明によれば、コンピュータ上でブートさせるＯＳを、第１のＯＳから第１のＯＳと異なる第２のＯＳへ切り替えることを容易に実現することができると共に、リモートでブートさせるための特別なプロトコルに依存することなく本発明を実施できるので汎用性にも優れている。

【００２１】ところで、第２のＯＳをブートさせるための情報に、第２のＯＳのブート・イメージ・ファイルが含まれている場合、コンピュータ上で第２のＯＳをブートさせることは、第２のＯＳのブート・イメージ・ファイルをコンピュータの主記憶装置に書き込み、第１のＯＳが終了した後に、ファームウェアのブート・ストラップ・コードによって主記憶装置の第２のＯＳのブート・イメージ・ファイルを書き込んだ領域がアクセスされるようにすることによって実現できる。

【００２２】また、ファームウェアのブート・ストラップ・コードによって主記憶装置の第２のＯＳのブート・イメージ・ファイルを書き込んだ領域がアクセスされるようにすることは、二次記憶装置Ｉ／Ｏサービス・コードをフックすること（ジャンプ先が登録されたテーブルを参照してテーブルに登録されているジャンプ先へジャンプする間接ジャンプにおいて、テーブルを書き替えてジャンプ先を変更すること）により実現できる。これにより、ファームウェアのブート・ストラップ・コードを

変更することなく第２のＯＳをブートすることができる。

【００２３】また、主記憶装置の各記憶領域をどのような用途に用いるかはＯＳによって異なっている。このため、本発明において、例えば第１のＯＳの終了を検知する前等の時期には、第２のＯＳをブートさせるための情報を、主記憶装置の記憶領域のうち第１のＯＳの制御下で上書きされない領域に一旦書き込んでおき、例えば第１のＯＳの終了を検知しかつ第２のＯＳをブートさせる前等のタイミングで、前記書き込んだ情報の一部又は全部（少なくとも現時点以降の処理で使用される情報）を、主記憶装置の記憶領域のうち少なくとも第２のＯＳのブートによって上書きされない領域に再配置することが好ましい。上記のように情報の再配置を行うことにより、再配置した情報が第２のＯＳのブートによって上書きされて破壊されることを回避することができ、第２のＯＳを正常にブートすることができる。

【００２４】なお、情報を再配置する領域は、主記憶装置の記憶領域のうち、第２のＯＳのブートによって上書きされず、かつ第２のＯＳがブートされた後の第２のＯＳの制御下でも上書きされない領域であってもよいし、第２のＯＳがブートされた後には前記再配置する情報を用いない場合には、第２のＯＳのブートによって上書きされず、第２のＯＳがブートされた後の第２のＯＳの制御下で上書きされる領域に前記情報を再配置するようにしてもよい。

【００２５】また本発明において、コンピュータ上で第２のＯＳをブートさせた後に、第２のＯＳが稼動している状態で実行させるべき処理（プログラム）としては、例えばファームウェアを更新するプログラム、特別なプロトコルを必要とするネットワーク・ブートを行うプログラム、セットアップを行うプログラム、コンピュータ・システムの診断を行うプログラム、第１のＯＳの稼動状態では動作が保証されない、或いは実行に適さない処理を行うプログラムが挙げられる。

【００２６】また、本発明において、第１のＯＳが、例えば自動プログラム起動機能や自動ログオン機能等を備えている場合には、これらの機能を利用することにより、コンピュータ上での第１のＯＳのブートから、少なくともコンピュータ上での第２のＯＳのブートに至る一連の処理（又は第２のＯＳがブートした後に実行すべき処理の実行も含んだ一連の処理）を自動的に連続して行なわせることも可能である。これにより、コンピュータに一連の処理を行なわせるための利用者の作業を省力化することができる。

【００２７】また本発明は、コンピュータ上でブートさせるＯＳを２種類のＯＳの中で切り替えることに限定されるものではなく、例えばＯＳ＃１が稼動している状態のコンピュータに対して本発明を適用し、前記コンピュータ上でＯＳ＃２をブートさせた後に、ＯＳ＃２が稼動している状態の前記コンピュータに対し、ＯＳ＃２を本発明に係る第１のＯＳとして本発明を適用し、前記コンピュータ上でＯＳ＃３をブートさせることも可能であり、上記を繰り返すことにより、コンピュータ上でブートさせるＯＳを３種類以上のＯＳの中で切り替えることも可能である。

【００２８】また、コンピュータ上でブートさせるＯＳを３種類以上のＯＳの中で切り替えることは、例えば第１のＯＳと異なる第２のＯＳをブートさせるための情報として、互いに異なる複数種のＯＳをブートさせるための情報を各々取得し、ＯＳの終了（第１のＯＳ又は他のＯＳの終了）を検知する毎に、前記複数種のＯＳのうちブート未実行のＯＳを前記コンピュータ上でブートさせることによっても実現できる。

【００２９】更に本発明において、第２のＯＳをブートさせるための情報を取得する前、又は取得した後に、コンピュータの利用者が第２のＯＳの正当な利用者か否かを確認することが好ましい。正当な利用者か否かの確認は、例えば利用者にパスワードを入力させ、入力されたパスワードを予め登録されたパスワードと照合することで行うことができる。これにより、正当な利用者以外の他者が、第２のＯＳをブートさせるための情報を取得してコンピュータ上で第２のＯＳをブートしたり、第２のＯＳが稼動している状態で所望の処理を実行させる等のように、コンピュータを不正に使用することを防止することができる。

【００３０】更に、第２のＯＳのアプリケーション・プログラムの中に、システムの構成によって動作させることができないものを含むことがある。そのような場合に、システム構成の一部又は識別ＩＤ等のシステムに関連する情報を利用者に入力させるか、システムから自動的に取得し、第２のＯＳのアプリケーション・プログラムと照合することにより、第２のＯＳのアプリケーション・プログラムを含む第２のＯＳのブート・イメージがコンピュータ上で誤ってブートされ、実行されてしまうことを防止することができる。

【００３１】また、第２のＯＳをブートさせるための情報が不正に改ざんされた情報でないことを認証するプロトコルを実施した後に第２のＯＳをブートすることにより、正当な利用者以外の他者がコンピュータを第２のＯＳの下で不正に使用することを防止することができる。

【００３２】また、本発明に係るコンピュータは、第１のＯＳが稼動している状態で、第１のＯＳと異なる第２のＯＳをブートさせるための情報が第１のＯＳの制御下で取得手段によって取得され、取得手段によって取得された情報が書込手段によってコンピュータの主記憶装置に書き込まれ、第１のＯＳの終了を検知すると、ブート制御手段により、書込手段によって主記憶装置に書き込まれた情報が消去されることなく、前記主記憶装置に書き込まれた情報によりコンピュータ上で第２のＯＳがブ

ートされるので、ブートさせるＯＳを容易に切り替え可能となる。

【００３３】また、本発明に係る記録媒体には、コンピュータ上で第１のＯＳが稼動している状態で、第１のＯＳと異なる第２のＯＳをブートさせるための情報を第１のＯＳの制御下で取得する第１のステップ、取得した情報をコンピュータの主記憶装置に書き込む第２のステップ、第１のＯＳの終了を検知すると、主記憶装置に書き込んだ情報を消去することなく、主記憶装置に書き込んだ情報によりコンピュータ上で第２のＯＳをブートさせる第３のステップを含む処理、すなわち本発明に係るコンピュータの制御方法をコンピュータによって実現するためのプログラムが記録されているので、コンピュータが前記記録媒体に記録されたプログラムを読み出して実行することにより、コンピュータ上でブートさせるＯＳを切り替えることを容易に実現できる。

【００３４】

【発明の実施の形態】以下、図面を参照して本発明の実施形態の一例を詳細に説明する。図１には、本発明を実現するのに適した典型的なパーソナル・コンピュータ（ＰＣ）から成るコンピュータ・システム１０のハードウェア構成がサブシステム毎に模式的に示されている。本発明を実現するＰＣの一例は、ＯＡＤＧ（PC Open Architecture Developer's Group）仕様に準拠し、オペレーティング・システム（ＯＳ）として米マイクロソフト社の”Ｗｉｎｄｏｗｓ９８又はＮＴ”又は米ＩＢＭ社の”ＯＳ／２”を搭載したノートブック型のＰＣ１２（図２参照）である。以下、コンピュータ・システム１０の各部について説明する。

【００３５】コンピュータ・システム１０全体の頭脳であるＣＰＵ１４は、ＯＳの制御下で、各種プログラムを実行する。ＣＰＵ１４は、例えば米インテル社製のＣＰＵチップ”Ｐｅｎｔｉｕｍ”、”ＭＭＸテクノロジＰｅｎｔｉｕｍ”、”Ｐｅｎｔｉｕｍ　Ｐｒｏ”や、ＡＭＤ社等の他社製のＣＰＵでも良いし、ＩＢＭ社製の”ＰｏｗｅｒＰＣ”でも良い。ＣＰＵ１４は、頻繁にアクセスするごく限られたコードやデータを一時格納することで、メイン・メモリ１６への総アクセス時間を短縮するための高速動作メモリであるＬ２（レベル２）－キャッシュを含んで構成されている。Ｌ２－キャッシュは、一般にＳＲＡＭ（スタティックＲＡＭ）チップで構成され、その記憶容量は例えば５１２ｋＢ又はそれ以上である。

【００３６】ＣＰＵ１４は、自身の外部ピンに直結されたプロセッサ直結バスとしてのＦＳＢ１８、高速のＩ／Ｏ装置用バスとしてのＰＣＩ（Peripheral Component Interconnect）バス２０、及び低速のＩ／Ｏ装置用バスとしてのＩＳＡ（Industry Standard Architecture）バス等から成るＩ／Ｏバス２２という３階層のバスを介して、後述の各ハードウェア構成要素と相互接続されている。

【００３７】ＦＳＢ１８とＰＣＩバス２０は、一般にメモリ／ＰＣＩ制御チップ２４と呼ばれるブリッジ回路（ホスト－ＰＣＩブリッジ）によって連絡されている。本実施形態のメモリ／ＰＣＩ制御チップ２４は、メイン・メモリ１６へのアクセス動作を制御するためのメモリ・コントローラ機能や、ＦＳＢ１８とＰＣＩバス２０の間のデータ転送速度の差を吸収するためのデータ・バッファ等を含んだ構成となっており、例えばインテル社製の４４０ＥＸや４４０ＧＸ等を用いることができる。

【００３８】メイン・メモリ１６は、ＣＰＵ１４の実行プログラムの読み込み領域として、或いは実行プログラムの処理データを書き込む作業領域として利用される書き込み可能メモリである。メイン・メモリ１６は、一般には複数個のＤＲＡＭ（ダイナミックＲＡＭ）チップで構成され、例えば３２ＭＢを標準装備し２５６ＭＢまで増設可能である。近年では、更に高速化の要求に応えるべく、ＤＲＡＭは高速ページＤＲＡＭ、ＥＤＯ　ＤＲＡＭ、シンクロナスＤＲＡＭ（ＳＤＲＡＭ）、バーストＥＤＯ　ＤＲＡＭ、ＲＤＲＡＭ等へと変遷している。

【００３９】なお、ここでいう実行プログラムには、Ｗｉｎｄｏｗｓ９８等のＯＳ、周辺機器類をハードウェア操作するための各種デバイス・ドライバ、特定業務に向けられたアプリケーション・プログラムや、フラッシュＲＯＭ５６（詳細は後述）に格納されたＢＩＯＳ（Basic Input/Output System：キーボードやフロッピーディスク・ドライブ等の各ハードウェアの入出力操作を制御するためのプログラム）等のファームウェアが含まれる。

【００４０】ＰＣＩバス２０は、比較的高速なデータ伝送が可能なタイプのバス（例えばバス幅３２／６４ビット、最大動作周波数３３／６６／１００MH_z、最大データ転送速度１３２／２６４ＭＢｐｓ）であり、カードバス・コントローラ３０のような比較的高速で駆動するＰＣＩデバイス類がこれに接続される。なお、ＰＣＩアーキテクチャは、米インテル社の提唱に端を発したものであり、いわゆるＰｎＰ（プラグ・アンド・プレイ）機能を実現している。

【００４１】ビデオ・サブシステム２６は、ビデオに関連する機能を実現するためのサブシステムであり、ＣＰＵ１４からの描画命令を実際に処理し、処理した描画情報をビデオメモリ（ＶＲＡＭ）に一旦書き込むと共に、ＶＲＡＭから描画情報を読み出して液晶ディスプレイ（ＬＣＤ）２８（図２参照）に描画データとして出力するビデオ・コントローラを含む。また、ビデオ・コントローラは、付設されたデジタル－アナログ変換器（ＤＡＣ）によってデジタルのビデオ信号をアナログのビデオ信号へ変換することができる。アナログのビデオ信号は、信号線を介してＣＲＴポート（図示省略）へ出力される。

【００４２】また、ＰＣＩバス２０にはカードバス・コントローラ３０、オーディオ・サブシステム３２及びモデム・サブシステム３４が各々接続されている。カードバス・コントローラ３０は、ＰＣＩバス２０のバス・シグナルをＰＣＩカードバス・スロット３６のインタフェース・コネクタ（カードバス）に直結させるための専用コントローラである。カードバス・スロット３６には、例えばＰＣ１２本体の壁面に配設され、ＰＣＭＣＩＡ（Personal Computer Memory Association）／ＪＥＩＤＡ（Japan Electronic Industry Development Association）が策定した仕様（例えば"PC Card standard 95"）に準拠したＰＣカード（図示せず）が装填される。

【００４３】また、モデム・サブシステム３４にはＬＡＮや電話回線等の通信回線が接続される。コンピュータ・システム１０はこれらの通信回線を介してインターネットに接続可能とされている。

【００４４】ＰＣＩバス２０とＩ／Ｏバス２２は多機能ＰＣＩデバイス３８によって相互に接続されている。多機能ＰＣＩデバイス３８は、ＰＣＩバス２０とＩ／Ｏバス２２とのブリッジ機能、ＤＭＡコントローラ機能、プログラマブル割り込みコントローラ（ＰＩＣ）機能、及びプログラマブル・インターバル・タイマ（ＰＩＴ）機能、ＩＤＥ（Integrated Drive Electronics）インタフェース機能、ＵＳＢ（Universal Serial Bus）機能、ＳＭＢ（System Management Bus）インタフェース機能を備えており、例えばインテル社製のＰＩＩＸ４というデバイスを用いることができる。

【００４５】なお、ＤＭＡコントローラ機能は、周辺機器（たとえばＦＤＤ）とメイン・メモリ１６との間のデータ転送をＣＰＵ１４の介在なしに実行するための機能である。またＰＩＣ機能は、周辺機器からの割り込み要求（ＩＲＱ）に応答して所定のプログラム（割り込みハンドラ）を実行させる機能である。また、ＰＩＴ機能はタイマ信号を所定周期で発生させる機能であり、その発生周期はプログラマブルである。

【００４６】また、ＩＤＥインタフェース機能によって実現されるＩＤＥインタフェースには、ＩＤＥハードディスク・ドライブ（ＨＤＤ）４０が接続される他、ＩＤＥＣＤ－ＲＯＭドライブ４２がＡＴＡＰＩ（AT Attachment Packet Interface）接続される。また、ＩＤＥ　ＣＤ－ＲＯＭドライブ４２の代わりに、ＤＶＤ（Digital Video Disc又はDigital Versatile Disc）ドライブのような他のタイプのＩＤＥ装置が接続されていても良い。ＨＤＤ４０やＣＤ－ＲＯＭドライブ４２等の外部記憶装置は、例えばＰＣ１２本体内の「メディア・ベイ」又は「デバイス・ベイ」と呼ばれる収納場所に格納される。これら標準装備された外部記憶装置は、ＦＤＤやバッテリ・パックのような他の機器類と交換可能かつ排他的に取り付けられる場合もある。

【００４７】なお、メイン・メモリ１６は本発明に係る主記憶装置に対応しており、ＨＤＤ４０は本発明に係る二次記憶装置に対応している。

【００４８】また、多機能ＰＣＩデバイス３８にはＵＳＢポートが設けられており、このＵＳＢポートは、例えばＰＣ１２本体の壁面等に設けられたＵＳＢコネクタ４４と接続されている。ＵＳＢは、電源投入のまま新しい周辺機器（ＵＳＢデバイス）を抜き差しする機能（ホット・プラギング機能）や、新たに接続された周辺機器を自動認識しシステム・コンフィギュレーションを再設定する機能（プラグ・アンド・プレイ）機能）をサポートしている。１つのＵＳＢポートに対して、最大６３個のＵＳＢデバイスをディジーチェーン接続することができる。ＵＳＢデバイスの例は、キーボード、マウス、ジョイスティック、スキャナ、プリンタ、モデム、ディスプレイモニタ、タブレットなど様々である。

【００４９】更に、多機能ＰＣＩデバイス３８にはＳＭバスを介してＥＥＰＲＯＭ５０が接続されている。ＥＥＰＲＯＭ５０はユーザによって登録されたパスワードやスーパーバイザー・パスワード、製品シリアル番号等の情報を保持するためのメモリであり、不揮発性で記憶内容を電気的に書き替え可能とされている。

【００５０】Ｉ／Ｏバス２２は、ＰＣＩバス２０よりもデータ転送速度が低いバスであり（例えばバス幅１６ビット、最大データ転送速度４ＭＢｐｓ）、Ｓｕｐｅｒ　Ｉ／Ｏコントローラ４６、電源コントローラ４８、ＥＥＰＲＯＭ等から成るフラッシュＲＯＭ５６、ＣＭＯＳ５８に加え、リアルタイム・クロック（ＲＴＣ）や、キーボード／マウス・コントローラのような比較的低速で動作する周辺機器類（何れも図示省略）を接続するのに用いられる。

【００５１】Ｓｕｐｅｒ　Ｉ／Ｏコントローラ４６にはＩ／Ｏポート５２が接続されており、フロッピーディスク・ドライブ（ＦＤＤ）の駆動、パラレル・ポートを介したパラレル・データの入出力（ＰＩＯ）、シリアル・ポートを介したシリアル・データの入出力（ＳＩＯ）を制御するための周辺コントローラである。

【００５２】電源コントローラ４８は主にコンピュータ・システム１０のパワー・マネージメントやサーマル・マネージメントを行うものであり、ＭＰＵ，ＲＡＭ，ＲＯＭ及びタイマ等を備えたシングルチップ・マイコンで構成することができる。ＲＯＭにはパワー・マネージメントやサーマル・マネージメントを実行するのに必要なプログラム及び参照テーブルが格納されている。電源コントローラ４８にはパワー・サプライ・コントローラ５４が接続されている。パワー・サプライ・コントローラ５４には、バッテリを充電するための充電器、コンピュータ・システム１０で使用される５Ｖ，３．３Ｖ等の直流定電圧を生成するためのＤＣ／ＤＣコンバータが含まれ、電源コントローラ４８の下で電力制御を行う。

【００５３】フラッシュＲＯＭ５６は、ＢＩＯＳやブート・ストラップ・コード等のファームウェアのプログラムを保持するためのメモリであり、不揮発性で記憶内容を電気的に書き替え可能とされている。また、ＣＭＯＳ５８は揮発性の半導体メモリがバックアップ電源に接続されて構成されており、不揮発性でかつ高速の記憶手段として機能する。

【００５４】なお、コンピュータ・システム１０を構成するためには、図１に示した以外にも多くの電気回路が必要である。但し、これらは当業者には周知であり、また、本発明の要旨を構成するものではないので、本明細書中では説明を省略する。また、図面の錯綜を回避するため、図中の各ハードウェアブロック間の接続も一部しか図示していないことを付記しておく。

【００５５】次に本実施形態の作用として、まず、コンピュータ・システム１０の電源スイッチがオンされることによって実行される第１のＯＳのブートについて、図３のフローチャートを参照して説明する。なお、第１のＯＳとしては、例えば米マイクロソフト社の”Ｗｉｎｄｏｗｓ９８又はＮＴ”や米ＩＢＭ社の”ＯＳ／２”のようなファイル・システムを備えたＯＳを適用することができる。

【００５６】電源スイッチがオンされると、フラッシュＲＯＭ５６の記憶領域のうち、ＢＩＯＳの一部であるＰＯＳＴ（Power On Self Test）のプログラムが記憶されている領域がアクセスされ、ＰＯＳＴのプログラムが実行される。これにより、図３に「システムテスト・初期化コード」として示すように、コンピュータ・システム１０の各ハードウェアがテストされると共にメイン・メモリ１６の初期化（記憶内容のクリア）が行われ（ステップ１００）、続いてコンピュータ・システム１０のハードウェア環境の初期化（具体的には外部ハードウェア割り込みベクタの初期化、外部ハードウェアの初期化、ソフトウェア割り込みベクタの初期化等）が行われる（ステップ１０２）。

【００５７】また、メイン・メモリ１６には、ＰＯＳＴ等を含むＢＩＯＳ、ブート・ストラップ・コード、二次記憶装置Ｉ／Ｏサービス等のファームウェアを記憶するための領域（システムＢＩＯＳ領域：例として図９参照）が設けられている。ＰＯＳＴは、これらのファームウェアをフラッシュＲＯＭ５６からメイン・メモリ１６のシステムＢＩＯＳ領域にコピーした後に、図３に「ブートストラップコードへジャンプ」と記されているように、メイン・メモリ１６上のシステムＢＩＯＳ領域のうち、ブート・ストラップ・コードが記憶されている領域へジャンプする。

【００５８】これにより、ブート・ストラップ・コードが実行され、まず予め定められた優先順位に従って複数のブート・ドライブ（ＯＳのブート・イメージが格納されている可能性のある二次記憶装置のドライブ（ＦＤＤやＨＤＤ４０等）：通常Ａドライブ（ＦＤＤ）が最優先）が順にアクセスされ、ブートすべきＯＳのブート・イメージが探索される。本実施形態では第１のＯＳのブート・イメージがＨＤＤ４０に記憶されており、ＦＤＤにＦＤがセットされていなければ、ブートすべきＯＳのブート・イメージとして、ＨＤＤ４０に記憶されている第１のＯＳのブート・イメージが発見される。そして、第１のＯＳのブート・イメージのうち、第１のＯＳをメイン・メモリ１６にロードするための一連の命令コード（第１のＯＳのローダ：第１のＯＳのブート・イメージの先頭に位置している）が、ＨＤＤ４０からメイン・メモリ１６へロードされる（ステップ１０６）。

【００５９】ブート・ストラップ・コードによる処理が終了すると、図３に「第１のＯＳのローダへジャンプ」と記されているように、メイン・メモリ１６の記憶領域のうち第１のＯＳのローダをロードした領域の先頭へジャンプし、第１のＯＳのローダのプログラム（一連の命令コード）が実行される。これにより、第１のＯＳのブート・イメージのうち、第１のＯＳ本体に相当する一連の命令コードが、ＨＤＤ４０からメイン・メモリ１６へロードされる（ステップ１１０）。

【００６０】第１のＯＳのローダによる処理が終了すると、図３に「第１のＯＳのスタートへジャンプ」と記されているように、メイン・メモリ１６の記憶領域のうちローダとＯＳによって予め取り決められた第１のＯＳのスタート番地へジャンプし、第１のＯＳのプログラム（一連の命令コード）が実行される。これにより第１のＯＳがブートされ、コンピュータ・システム１０上で第１のＯＳが稼動している状態となる。

【００６１】次に、第１のＯＳが稼動している状態から、第１のＯＳと異なる第２のＯＳをブートさせ、第２のＯＳの制御下で所定のアプリケーションを実行させる一連の処理について、図４（及び図５）のフローチャートを参照して説明する。なお、以下では第２のＯＳの一例として、ＤＯＳを適用した場合について説明する。

【００６２】図４（及び図５）に示す一連の処理は、システム又はユーザがＯＳ切り替え起動プログラムを起動することにより実行を開始する。なお、ＯＳ切り替え起動プログラムは、例えば後述するステップ１３０と同様にしてサーバからダウンロードすることで取得することができる。また、ＯＳ切り替え起動プログラムは、第１のＯＳの制御下で実行されるアプリケーション・プログラムとして予めＨＤＤ４０等にインストールされていてもよい。

【００６３】ステップ１３０では、第２のＯＳをブートし第２のＯＳの制御下で所定のアプリケーションを実行するための情報を、第１のＯＳの制御下で、ネットワーク（例えばインターネット）を経由してサーバからダウンロードする。なお、本実施形態において、第２のＯＳをブートするための情報は、第２のＯＳのブート・イメ

ージ・ファイル、デバイス・ドライバ、及び二次記憶装置Ｉ／Ｏバッチ・プログラムから構成されており、第２のＯＳのブート・イメージ・ファイルには、第２のＯＳのローダ、第２のＯＳ本体、及び第２のＯＳの制御下で実行すべき所定のアプリケーション・プログラムが含まれている。

【００６４】ダウンロードの方法としては種々の方法があり、例えばコンピュータ・システム１０の利用者が閲覧ソフト（ブラウザ）等を起動し、サーバによって提供されるダウンロードを行うためのホームページを閲覧することでダウンロードする方法がある。一例として図６に示すホームページでは、第２のＯＳの制御下で実行可能でダウンロード可能な複数種のアプリケーション・プログラム（図６の例ではコンピュータ・システム１０に対してベンチマーク・テストを行うためのプログラム、ＢＩＯＳの更新を行うためのプログラム、及びコンピュータ・システム１０の自己診断を行うためのプログラムの３種類）の名称が各々表示されている。

【００６５】この場合、一例として複数種のアプリケーション・プログラムの名称の中から、利用者が実行を所望しているアプリケーション・プログラムの名称が利用者によって選択されると、まずＯＳ切り替え起動プログラムがサーバからコンピュータ・システム１０にバイナリ・ファイルとしてダウンロードされ、コンピュータ・システム１０上でＯＳ切り替え起動プログラムが起動されてステップ１３０が実行されることにより、選択されたアプリケーション・プログラムを含む第２のＯＳのブート・イメージ・ファイル、デバイス・ドライバ、二次記憶装置Ｉ／Ｏバッチ・プログラムが順にバイナリ・ファイルとしてダウンロードされるようにすることができる。

【００６６】また、他のダウンロードの方法として、例として図７に示すように、第２のＯＳのブート・イメージ・ファイル、デバイス・ドライバ、二次記憶装置Ｉ／Ｏバッチ・プログラム（及びＯＳ切り替え起動プログラム）を電子メールの添付ファイルとして受信する方法がある。

【００６７】図７の例では、送信者により、ＯＳ切り替え起動プログラム（図７では「doboot.exe」と表記）、デバイス・ドライバ（図７では「dosboot.sys」及び「DOSBOOT.vxd」と表記）、二次記憶装置Ｉ／Ｏバッチ・プログラム（図７では「dosboot.com」と表記）、及びＢＩＯＳを更新するアプリケーション・プログラムを含む第２のＯＳのブート・イメージ・ファイル（図７では「biosupdt.img」と表記）がバイナリ・ファイルとして各々添付されて送信された電子メールがコンピュータ・システム１０で受信され、受信した電子メールを開いた状態の画面イメージが示されている。これらのバイナリ・ファイルを一旦二次記憶装置の仮のディレクトリにダウンロードし、「doboot.exe」と表記されたＯＳ切り替え起動プログラムのアイコンをクリックすればＯＳ切り替え起動プログラムがメイン・メモリ１６にロードされて次のステップ１３２以降の処理が実行されることになる。

【００６８】なお、上述した各種のダウンロードは、何れも第１のＯＳの制御下で行われるダウンロードであり、リモートでブートさせるための特別なプロトコルに依存することなく、汎用的なプロトコルによってダウンロードを行うことができる。また、ダウンロードされたデバイス・ドライバ、二次記憶装置Ｉ／Ｏバッチ・プログラム、及び第２のＯＳのブート・イメージ・ファイルは、第１のＯＳによってファイルとして認識され、第１のＯＳのファイル・システムの管理下でＨＤＤ４０に一旦記憶される。

【００６９】次のステップ１３２では、例として図８に示すようなパスワード入力画面をＬＣＤ２８に表示する等により、利用者に対してパスワードの入力が要請される。パスワードが一致していなかった場合には、ステップ１３４の判定が否定されてＯＳ切り替え起動プログラムによる処理を終了する。この場合、次のステップ１３６以降の処理は行われず、第２のＯＳがブートされることはないので、正当な利用者以外の他者によるコンピュータ・システム１０の不正使用を防止することができる。なお、正当な使用者か否かの確認は上記のタイミングで行うことに限定されるものではなく、例えば第１のＯＳのファイル・システムによって管理されているデバイス・ドライバや二次記憶装置Ｉ／Ｏバッチ・プログラム、第２のＯＳのブート・イメージ・ファイルがアクセスされたときに行うようにしてもよい。

【００７０】また、パスワードが一致していた場合には、ステップ１３４の判定が肯定されてステップ１３６へ移行し、ＨＤＤ４０にファイルとして記憶されているデバイス・ドライバ（一連の命令コードから成るバイナリ・ファイル）をメイン・メモリ１６に常駐コードとしてロード（詳しくは、第１のＯＳの終了時にフックしたコードから制御を受け取る（詳細は後述）ために、二次記憶装置に退避されない常にメイン・メモリ１６に存在するコードとしてロードする）した後に、デバイス・ドライバを起動させる。起動されたデバイス・ドライバは、ＯＳ切り替え起動プログラムからＯＳ終了のフックが要求される迄待機する（ステップ１４４）。

【００７１】ＯＳ切り替え起動プログラムは、デバイス・ドライバを起動させると、次のステップ１３８でデバイス・ドライバに対してＯＳ終了のフックを要求する。これにより、デバイス・ドライバは第１のＯＳの終了をフックする処理を行う。すなわち、第１のＯＳは、例えばコンピュータ・システム１０の再起動や電源オフのために終了処理の実行が指示されると、起動されているアプリケーション・プログラムを全て終了させ、開かれている全てのファイルをクローズし、必要に応じてデータ

を保存する一連のＯＳ終了処理を行い、ＯＳ終了処理が完了した後にコンピュータ・システムの再起動又は電源オフを行うプログラム（これもＯＳの一部）にジャンプし、コンピュータ・システムの再起動又は電源オフを行う。

【００７２】コンピュータ・システムの再起動（又は電源オフ）を行うプログラムへのジャンプは、ジャンプ先が登録されたテーブルを参照して、テーブルに登録されているジャンプ先にジャンプする間接ジャンプである。デバイス・ドライバは、前記テーブルに登録されているジャンプ先を書き替えることによって第１のＯＳの終了をフックし、ＯＳ終了処理が完了した後に、ＣＰＵ１４の制御がコンピュータ・システムの再起動（又は電源オフ）を行うプログラムへジャンプする代わりに、メイン・メモリ１６の記憶領域のうちデバイス・ドライバがロードされている領域にジャンプし、自ら（デバイス・ドライバ）に移るようにする。

【００７３】また、デバイス・ドライバは第１のＯＳの終了をフックする処理を行うと、次のステップ１４８において、第１のＯＳに対してメイン・メモリ１６の記憶領域にワーク領域を確保するよう依頼し、ＨＤＤ４０にファイルとして記憶されている二次記憶装置Ｉ／Ｏバッチ・プログラム、第２のＯＳのブート・イメージ・ファイル（何れも一連の命令コードから成るバイナリ・ファイル）を、第１のＯＳによって確保されたワーク領域（第１のＯＳの制御下で上書きされない領域）にロードする（図９（Ａ）の「二次記憶装置Ｉ／Ｏバッチプログラム」及び「ＤＯＳブートイメージ」を参照）。そしてデバイス・ドライバは、ステップ１４８を実行後に処理を中断する。

【００７４】一方、ＯＳ切り替え起動プログラムでは、デバイス・ドライバに対してＯＳ終了のフックを要求すると、次のステップ１４０において、第１のＯＳに対してＯＳ終了を要求する。このＯＳ終了要求は、ＯＳ切り替え起動プログラム自身が第１のＯＳの終了を指示するＯＳのサービスを呼び出すか、又は利用者のインタラクティブな操作によって第１のＯＳの終了が指示される（例えば利用者に対し「現在稼動中のＯＳの終了を指示すれば、ダウンロードしたアプリケーションが実行されます」等の案内メッセージをＬＣＤ２８に表示し、この案内メッセージに従って利用者がＯＳの終了を指示する等）ことによって行われる。上記のようにＯＳの終了を要求すると、ＯＳ切り替え起動プログラムは処理を終了する。

【００７５】稼動中の第１のＯＳは、ＯＳ終了要求が有る（ステップ１５８の判定が肯定される）迄の間は通常の処理（ステップ１５６）を行うが、ＯＳ切り替え起動プログラムからＯＳ終了が指示されるとステップ１５８の判定が肯定され、通常の手順でコンピュータ・システム１０の再起動や電源オフが指示された場合と同様に、起動されているアプリケーション・プログラムを全て終了させ、開かれている全てのファイルをクローズし、必要に応じてデータを保存する一連のＯＳ終了処理を行う（ステップ１６０）。

【００７６】ここで、通常の手順でコンピュータ・システム１０の再起動や電源オフが指示された場合には、ＯＳ終了のフック（ステップ１４６）が行われていない（ステップ１６２の判定が否定）ので、コンピュータ・システムの再起動又は電源オフを行うプログラムにジャンプし、コンピュータ・システムの再起動又は電源オフが行われる。しかし、ＯＳ終了のフックが行われていた場合（ステップ１６２の判定が肯定される）には、ＯＳ終了処理が完了するとＣＰＵ１４の制御がデバイス・ドライバに移ることでデバイス・ドライバの処理が再開される。

【００７７】ところで，本実施形態において第２のＯＳとして用いているＤＯＳは、プロセッサ（ＣＰＵ１４）のモードがリアル・モードの状態で動作するのに対し、第１のＯＳとして適用可能な他のＯＳ（本実施形態において第１のＯＳの一例として挙げた”Ｗｉｎｄｏｗｓ９８又はＮＴ”や”ＯＳ／２”、或いはその他のＯＳ）はプロセッサのモードがリアル・モード以外のモードになっている状態で動作する。このため、第２のＯＳをブートするにあたってはプロセッサのモードを変更する必要がある。しかし、プロセッサのモードを変更すると、プロセッサの仮想記憶機構が有効になったり無効になったりすることがある（一例としてプロセッサのモードを、”Ｗｉｎｄｏｗｓ”等のＯＳが動作可能なモードからリアル・モードへ変更した場合、仮想記憶機構は有効→無効に変化する）。

【００７８】従って、再起動されたデバイス・ドライバは、次のステップ１５０において、プロセッサのモードを変更するための前処理として、プロセッサのモードを第２のＯＳが動作可能なモード（本実施形態ではリアル・モード）へ変更するための一連のプログラム・コード（プロセッサのモードを遷移させる一連の命令コード）を、メイン・メモリ１６の記憶領域のうち、仮想アドレスと物理アドレスが同一の領域（仮想記憶機構が有効→無効（又はその逆）に変化しても同一の領域として認識される領域）にロードする。

【００７９】また、メイン・メモリ１６に既にロードした二次記憶装置Ｉ／Ｏバッチ・プログラム及び第２のＯＳのブート・イメージ・ファイルについても、第２のＯＳが動作するモードでアクセス可能である必要がある。このため、ステップ１５０では、メイン・メモリ１６の記憶領域上で物理的に連続している領域を確保し、二次記憶装置Ｉ／Ｏバッチ・プログラム及び第２のＯＳのブート・イメージ・ファイルを前記確保した領域に転送（再配置）する。

【００８０】上記の処理を行うと、メイン・メモリ１６

の記憶領域のうち、ステップ１５０で一連のプログラム・コードをロードした領域にジャンプし、前記一連のプログラム・コードを実行する。これにより、プロセッサ自身のモードが第２のＯＳが動作可能なモード（リアル・モード）へ遷移する（ステップ１５２）。なお、プロセッサのモードを遷移させる一連のプログラム・コードは仮想アドレスと物理アドレスが同一の領域にロードされているので、プロセッサのモードの遷移に拘わらず、一連のプログラム・コードを最後まで正常に実行することができる。

【００８１】一連のプログラム・コードを最後まで実行し、プロセッサのモードの変更が完了すると、図４に「バッチプログラムへジャンプ」と記されているように、メイン・メモリ１６の記憶領域のうち、先のステップ１５０で二次記憶装置Ｉ／Ｏバッチ・プログラムを転送した領域へジャンプし、二次記憶装置Ｉ／Ｏバッチ・プログラムが起動される。以下、二次記憶装置Ｉ／Ｏバッチ・プログラムによって実現される処理について、図５のフローチャートを参照して説明する。

【００８２】ステップ１７０では、あたかもコンピュータ・システム１０のファームウェアが第２のＯＳを直接ロードしたかのように（第２のＯＳが第１のＯＳとしてロードされるかのように）、ＢＩＯＳの一部であるＰＯＳＴに代わってハードウェア環境の初期化（具体的には外部ハードウェア割り込みベクタの初期化、外部ハードウェアの初期化、ソフトウェア割り込みベクタの初期化等）を行う。なお、ソフトウェア割り込みベクタの一部である二次記憶装置Ｉ／Ｏサービス割り込みベクタについては、後述するように再初期化される。

【００８３】また、第２のＯＳをブートするためには、メイン・メモリ１６への第２のＯＳ本体のロードにより、メイン・メモリ１６に記憶されている第２のＯＳのブート・イメージ・ファイル及び二次記憶装置Ｉ／Ｏバッチ・プログラムが破壊（上書き）されないようにする必要がある。このため、次のステップ１７２では、第２のＯＳのブート・イメージ・ファイル及び二次記憶装置Ｉ／Ｏバッチ・プログラム自身を、第２のＯＳのロードによって上書きされない領域（第２のＯＳ本体がロードされず、第２のＯＳのワーク領域として使用されることもない領域）へ転送する。

【００８４】具体的には、例えば第２のＯＳとしてのＤＯＳの制御下では、図９に示すように、メイン・メモリ１６の記憶領域のうち、アドレスE0000h（「ｈ」は１６進数表示であることを表す）を先頭とする所定サイズの領域はＰＯＳＴ等を含むＢＩＯＳ、ブート・ストラップ・コード、二次記憶装置Ｉ／Ｏサービス等のファームウェアの命令コードを記憶するための領域（システムＢＩＯＳ領域）として用いられ、アドレス400hを先頭とする所定サイズの領域は前記ファームウェアのデータを記憶するための領域（ＢＤＡ：ＢＩＯＳデータ領域）として用いられる。

【００８５】また、メイン・メモリ１６の記憶領域には、ＢＩＯＳデータ領域と別に、ファームウェアのデータを記憶するための第２の領域（ＥＢＤＡ＝Extended BIOS Data Area：拡張ＢＩＯＳデータ領域）も設けられている。この拡張ＢＩＯＳデータ領域の先頭アドレスはＢＩＯＳデータ領域の中のアドレス40Eh，40Fhにデータとして記憶されている（アドレス40Eh，40Fhに記憶されているデータは通常はセグメントＸを指している：図９（Ａ）参照）。

【００８６】このためステップ１７２では、拡張ＢＩＯＳデータ領域がセグメントＸよりも低アドレス（詳しくは二次記憶装置Ｉ／Ｏバッチ・プログラムを記憶するのに必要なサイズ分だけ低アドレス）のセグメントＹから始まるように、ＢＩＯＳデータ領域のアドレス40Eh，40Fhに記憶されているデータを書き替え（これにより拡張ＢＩＯＳデータエリアのサイズが拡大される）、サイズを拡大した拡張ＢＩＯＳデータエリアの末尾側の空き領域に二次記憶装置Ｉ／Ｏバッチ・プログラム自身を転送する（図９（Ｂ）参照）。

【００８７】また、ＢＩＯＳデータ領域の中のアドレス413h，414hには、メイン・メモリ１６の記憶領域のうち、アドレス０ｈから１Ｍバイトの間の領域における空き領域のサイズを表すデータが格納されており、上記のように拡張ＢＩＯＳデータエリアのサイズを拡大したことに伴い、アドレス413h，414hに記憶されているデータも書き替える。これにより、拡大した拡張ＢＩＯＳデータ領域（二次記憶装置Ｉ／Ｏバッチ・プログラム自身を転送した領域を含む）にＤＯＳ本体がロードされる等により、二次記憶装置Ｉ／Ｏバッチ・プログラムが上書きされて破壊されることを防止できる。

【００８８】なお、第２のＯＳのブート・イメージ・ファイルについては、メイン・メモリ１６の先頭から例えば４Ｍバイト以上隔たった高位アドレスの領域に転送する。この領域は、搭載メモリサイズ・クエリー・ソフトウェア(memory size query software)割り込みハンドラのフック（詳細は後述する）により、ＤＯＳ本体がロードされたりＤＯＳのワーク領域として使用されることはないので、第２のＯＳのブート・イメージ・ファイルが上書きされて破壊されることも防止することができる。

【００８９】また、メイン・メモリ１６のアドレス０ｈを先頭とする所定サイズの領域には、ソフトウェア割り込み等が発生したときのジャンプ先が割り込み要因毎に記憶されたテーブル（割り込みベクタ）が記憶されている。割り込みベクタは、各種のソフトウェア割り込みのうち二次記憶装置Ｉ／Ｏサービス割り込み（割り込み要因＝ＩＮＴ13hの割り込み）が発生した場合に、システムＢＩＯＳ領域に記憶されているファームウェア（二次記憶装置Ｉ／Ｏサービス（ＩＮＴ13hハンドラ））へジャンプするように初期化され、ファームウェアの二次記憶

装置Ｉ／Ｏサービスを経由して指定された二次記憶装置がアクセスされる。

【００９０】これに対し、次のステップ１７４では、二次記憶装置Ｉ／Ｏサービス割り込みが発生した場合に、二次記憶装置Ｉ／Ｏバッチ・プログラムの一部であるＩＮＴ13hハンドラへジャンプするように割り込みベクタを書き替えることで、二次記憶装置Ｉ／Ｏサービス・ソフトウェア割り込みハンドラをフックする。

【００９１】二次記憶装置Ｉ／Ｏバッチ・プログラムのＩＮＴ13hハンドラは、割り込みベクタからジャンプすることで起動されると、まずプロセッサ内部のＤＬレジスタの値（ＤＬレジスタの値はアクセスする二次記憶装置のドライブ番号を表す）をチェックする。そして、ＤＬレジスタの値が、優先順位が最も高い二次記憶装置のドライブ（通常はＦＤＤ）からの読み出しサービスを表す値（DL＝０）のときには、図１０に示すように、メイン・メモリ１６の記憶領域のうち第２のＯＳのブート・イメージ・ファイルが記憶されている領域がアクセスされて読み出されるように制御する。

【００９２】一方、ＤＬレジスタの値が他の値（DL≠０）のときには、ファームウェアの二次記憶装置Ｉ／Ｏサービスへジャンプするように制御する。この場合、図１０に示すように、ＤＬレジスタによって指定された二次記憶装置のドライブがアクセスされることになる。

【００９３】また、一般にコンピュータ・システムは、ＯＳやＯＳで実行されるプログラムにより、搭載メモリ（メイン・メモリ１６）のサイズを問い合わせるソフトウェア割り込み（割り込み要因＝ＩＮＴ15hの割り込み）が発生されると、搭載メモリのサイズを通知するサービス（搭載メモリサイズのクエリー）を提供している。ＯＳやＯＳで実行されるプログラムの中には、初期化時に搭載メモリサイズ・クエリー・ソフトウェア割り込みを発生させて搭載メモリサイズを取得し、搭載メモリの上限アドレスまで領域を自身のワーク領域として使用することを要求するものがある。

【００９４】このため、次のステップ１７６では、搭載メモリサイズ・クエリー・ソフトウェア割り込みが発生した場合に、二次記憶装置Ｉ／Ｏバッチ・プログラムの一部であるＩＮＴ15hハンドラへジャンプするように割り込みベクタを書き替えることで、搭載メモリサイズ・クエリー・ソフトウェア割り込みハンドラをフックする。

【００９５】二次記憶装置Ｉ／Ｏバッチ・プログラムのＩＮＴ15hハンドラは、割り込みベクタからジャンプすることで起動されると、割り込み発生元に対して通知される搭載メモリサイズを、第２のＯＳのブート・イメージ・ファイルが記憶されている領域よりも低位のアドレス（例えばメイン・メモリ１６の先頭から４Ｍバイト隔てた領域のアドレス）迄しか搭載メモリの記憶領域が存在しないことを意味する値に変更する。これにより、割り込み発生元（ＯＳやＯＳで実行されるプログラム）からは、第２のＯＳのブート・イメージ・ファイルが記憶されている領域が実装されていないように見えることになり、第２のＯＳのブート・イメージ・ファイルが上書きされることが防止される。

【００９６】上記により二次記憶装置Ｉ／Ｏバッチ・プログラムによる処理が終了し、図５に「ブートストラップコードへジャンプ」と記されているように、メイン・メモリ１６のシステムＢＩＯＳ記憶領域のうち、ブート・ストラップ・コードが記憶されている領域へジャンプし、ブート・ストラップ・コードが実行される。

【００９７】これにより、まず優先順位が最も高いブート・ドライブにブートすべきＯＳのブート・イメージが存在しているか否かを確認するために、ＤＬレジスタの値が０にされて二次記憶装置Ｉ／Ｏサービス割り込み（ＩＮＴ13hの割り込み）が発生されるが、先に説明したように、二次記憶装置Ｉ／Ｏサービス割り込みは割り込みベクタの書き替えによってフックされているので、二次記憶装置Ｉ／Ｏバッチ・プログラムのＩＮＴ13hハンドラを経由して、メイン・メモリ１６の記憶領域のうち第２のＯＳのブート・イメージ・ファイルが記憶されている領域がアクセスされ、ブートすべきＯＳのブート・イメージとして第２のＯＳのブート・イメージが発見される。

【００９８】そして、第２のＯＳのブート・イメージのうち、第２のＯＳ本体をメイン・メモリ１６にロードするための一連の命令コード（第２のＯＳのローダ：第２のＯＳのブート・イメージの先頭に位置している）が、メイン・メモリ１６上の所定の領域にロードされる（ステップ１８０）。

【００９９】また、第２のＯＳのブート・イメージ・ファイルの先頭番地は二次記憶装置Ｉ／Ｏバッチ・プログラムにとって自明であるので、ブート・ストラップ・コードが実行する第２のＯＳのローダのロード（ステップ１８０）を、二次記憶装置Ｉ／Ｏバッチ・プログラムが直接メイン・メモリ１６上の第２のＯＳのブート・イメージ・ファイルの領域をアクセスし、メイン・メモリ１６の所定の領域に転送することによって実現してもよい。このようにすれば、ブート・ストラップ・コードの優先順位に拘わらず第２のＯＳをブートすることができる。

【０１００】ブート・ストラップ・コードによる処理が終了すると、図５に「第２のＯＳのローダへジャンプ」と記されているように、メイン・メモリ１６の記憶領域のうち第２のＯＳのローダをロードした領域の先頭へジャンプし、第２のＯＳのローダのプログラム（一連の命令コード）が実行される。これにより、第２のＯＳのブート・イメージのうち、第２のＯＳ本体に相当する一連の命令コードが、メイン・メモリ１６上の所定の領域（図９及び図１０に「ＤＯＳ本体及びワークエリア」と

表記している領域）にロードされる（ステップ１８２）。

【０１０１】第２のＯＳのローダによる処理が終了すると、図５に「第２のＯＳのスタートへジャンプ」と記されているように、メイン・メモリ１６の記憶領域のうちローダとＯＳによって予め取り決められた第２のＯＳのスタート番地へジャンプし、第２のＯＳのプログラム（一連の命令コード）が実行される。これにより第２のＯＳがブートされ、コンピュータ・システム１０上で第２のＯＳが稼動している状態となる。

【０１０２】また、第２のＯＳのブート・イメージ・ファイルには、第２のＯＳの制御下で実行すべき所定のアプリケーション・プログラムが含まれており、第２のＯＳがブートされると、所定のアプリケーション・プログラムがメイン・メモリ１６の所定の領域にロードされて実行され、所定の処理（例えばコンピュータ・システム１０に対するベンチマーク・テスト、ＢＩＯＳの更新、コンピュータ・システム１０の自己診断等）が行われる（ステップ１８６）。なお、このアプリケーション・プログラムのロード及び実行は、自動的に行うようにしてもよいし、利用者による指示に応じて行うようにしてもよい。

【０１０３】このように、本実施形態では、第２のＯＳをブートし第２のＯＳの制御下で所定のアプリケーションを実行するための情報をダウンロードすると共にＯＳ切り替え起動プログラムを起動すれば、後は自動的に第２のＯＳがブートされて所定のアプリケーションが実行されるので、コンピュータ上でブートさせるＯＳを切り替えを容易に実現できる。また、第１のＯＳの制御下で前記情報のダウンロードを行うので、リモートでブートさせるための特別なプロトコルに依存することなく、汎用的なプロトコルによってダウンロードを行うことができ、汎用性に優れている。

【０１０４】また、本実施形態における第２のＯＳのブートは、第２のＯＳのブート・イメージ・ファイルやその他のプログラムをメイン・メモリ１６に記憶して行うものであるので、コンピュータ・システム１０にＦＤＤが実装されていない場合にも適用可能であると共に、ハードディスク等の記憶領域を複数のパーティションに分割したり、マスタ・ブート・レコードを書き替える等の煩雑な作業も不要である。

【０１０５】更に、本実施形態では、第２のＯＳの制御下で所定のアプリケーション・プログラムを実行する際に、前記所定のアプリケーション・プログラムを含む情報をダウンロードするので、前記所定のアプリケーション・プログラムとして、最新のバージョンのプログラムを容易に入手することができる。

【０１０６】また、本実施形態では、所定のアプリケーションの実行が完了し、第２のＯＳが稼動している状態で、コンピュータ・システム１０の再起動を指示したり、或いはコンピュータ・システム１０の電源をオフした後に再度電源をオンする等の操作を行った場合には図３に示した通常のシーケンスが実行されるので、何ら特別な操作を行うことなく自動的に第１のＯＳを再ブートさせることができる。

【０１０７】なお、上記では第１のＯＳが稼動している状態から第２のＯＳをブートさせる場合について説明したが、本発明は上記に限定されるものではなく、３種類以上のＯＳを順にブートさせることも可能である。３種類以上のＯＳを順にブートさせる方法としては２つの方法が考えられる。

【０１０８】第１の方法は、順にブートする各ＯＳがネットワークを介して通信を行う機能を各々有している場合に適用可能な方法であり、図１１に示すように、Ｎ番目のＯＳが稼動している状態で、次（Ｎ＋１番目）のＯＳのブートに必要な情報をダウンロードし、Ｎ＋１番目のＯＳをブートすることを繰り返す方法である。この第１の方法では、Ｎ番目のＯＳからＮ＋１番目のＯＳへの切り替えの際に、Ｎ－１番目以前のＯＳによってダウンロードされた情報（この情報はローカルディスクやＮ－１番目以前のＯＳの管理下にあるメモリ領域に格納されている可能性もある）にアクセスする必要がないので、順にブートするＯＳが異なるファイル・システムを備えていたり、メモリアドレス空間が分離されていて複数のＯＳの間で共通の情報格納場所を確保できない環境である等の場合にも適用できるという利点がある。

【０１０９】また、第２の方法は、複数のＯＳの間で共通の情報格納場所を確保できる場合に適用可能な方法であり、図１２に示すように、Ｎ番目のＯＳが稼動している状態で、Ｎ＋１番目以降の複数のＯＳのブートに必要な情報を一括してダウンロードして二次記憶装置に各々記憶しておき、ＯＳの切り替え時に必要な情報を二次記憶装置から読み出す方法である（なお、図１２はＮ＝１の場合を示す）。この第２の方法では、順にブートする各ＯＳの中にネットワークを介して通信を行う機能を備えていないＯＳが含まれている場合（但し第１のＯＳ以外）にも適用できるという利点がある。

【０１１０】また、上記では第２のＯＳの制御下で実行可能な複数種のアプリケーション・プログラムのうち、単一のアプリケーション・プログラム（詳しくは単一のアプリケーション・プログラムのみを含む第２のＯＳのブート・イメージ・ファイル）をダウンロードする例を説明したが、これに限定されるものではなく、例えば第２のＯＳのブート・イメージ・ファイル、デバイス・ドライバ、二次記憶装置１／Ｏバッチ・プログラムと別に、複数種のアプリケーション・プログラムを全てダウンロードし、第２のＯＳが稼動している状態で、ダウンロードした複数種のアプリケーション・プログラムの中から、実行すべきアプリケーションを利用者に選択させるようにしてもよい。

【０１１１】更に、上記では正当な使用者か否かの確認をダウンロード後に行っていたが、これに限定されるものではなく、例えばホームページを閲覧して必要な情報をダウンロードする態様において、ダウンロードが要求された場合にサーバ側でパスワードやＩＰアドレスの照合等を行い、正当な使用者であることを確認した場合にのみダウンロードを許可するようにしてもよい。また、ホームページを閲覧して必要な情報をダウンロードする態様において、例えばＢＩＯＳの更新を行うアプリケーションのダウンロードが要求された場合に、要求元のコンピュータの機種や製造年月、或いはＢＩＯＳのバージョンをサーバが確認し、ＢＩＯＳの更新が必要と判断したコンピュータからのダウンロード要求のみを許可するようにしてもよい。

【０１１２】また、上記では本発明を実施するためのプログラム（ＯＳ切り替え起動プログラム、デバイス・ドライバ、及び二次記憶装置Ｉ／Ｏバッチ・プログラム）をネットワーク経由でダウンロードする態様を説明したが、本発明はこれに限定されるものではなく、本発明を実施するための全てのプログラムをフロッピーディスクやＣＤ－ＲＯＭ等の記録媒体に記録しておき、前記記録媒体（本発明に係る記録媒体に相当）から前記プログラムをコンピュータにインストールするようにしてもよい。これにより、前記プログラムをインストールしたコンピュータに対して本発明に係るコンピュータの制御方法を適用することが可能となる（前記コンピュータを本発明に係るコンピュータとして機能させることが可能となる）。

【０１１３】また、上記では本発明に係るコンピュータの一例としてノートブック型のＰＣ１２を例に説明したが、これに限定されるものではなく、デスクトップ型のＰＣや他のコンピュータであってもよいことは言うまでもない。また、本発明に係る第２のＯＳもＤＯＳに限定されるものではなく、公知の様々なＯＳを第２のＯＳとして採用可能であることは言うまでもない。

【０１１４】

【発明の効果】以上説明したように本発明は、コンピュータ上で稼動している第１のＯＳと異なる第２のＯＳをブートさせるための情報を第１のＯＳの制御下で取得し、取得した情報を前記コンピュータの主記憶装置に書き込み、第１のＯＳの終了を検知すると、主記憶装置に書き込んだ情報を消去することなく、主記憶装置に書き込んだ情報により第２のＯＳをブートさせるようにしたので、コンピュータ上でブートさせるＯＳを切り替えることを容易に実現でき、汎用性に優れている、という優れた効果を有する。

【０１１５】更に、既存のコンピュータにハードウェアやファームウェアの変更を加えることなく、ブートさせるＯＳを容易に切り替えることができ、利用範囲が広いという効果を有する。

【図面の簡単な説明】

【図１】　本実施形態に係るコンピュータ・システムの概略構成を示すブロック図である。

【図２】　ノートブック型ＰＣの外観を示す斜視図である。

【図３】　第１のＯＳをブートさせる処理の内容を示すフローチャートである。

【図４】　第１のＯＳが稼動している状態から第２のＯＳをブートさせるための前処理（ＯＳ切り替え起動プログラム及びデバイス・ドライバによる処理）の内容を示すフローチャートである。

【図５】　図４の処理に続いて、第２のＯＳをブートさせて所定のアプリケーションを実行させる一連の処理（二次記憶装置Ｉ／Ｏバッチ・プログラム等による処理）の内容を示すフローチャートである。

【図６】　ホームページを閲覧して、第２のＯＳをブートして所定のアプリケーション・プログラムを実行するための情報をダウンロードする場合の画面イメージの一例を示す図である。

【図７】　第２のＯＳをブートして所定のアプリケーション・プログラムを実行するための情報を電子メールの添付ファイルとして受信する場合の画面イメージの一例を示す図である。

【図８】　パスワードによって利用者を確認する際の画面イメージの一例を示す図である。

【図９】　（Ａ）は二次記憶装置Ｉ／Ｏバッチ・プログラム及び第２のＯＳのブート・イメージ・ファイルを転送する前、（Ｂ）は前記プログラム及びファイルを転送し第２のＯＳをブートした後のメイン・メモリのメモリマップを示すイメージ図である。

【図１０】　二次記憶装置Ｉ／Ｏサービス・ソフトウェア割り込みのフック、搭載メモリサイズ・クエリー・ソフトウェア割り込みのフックを説明するためのイメージ図である。

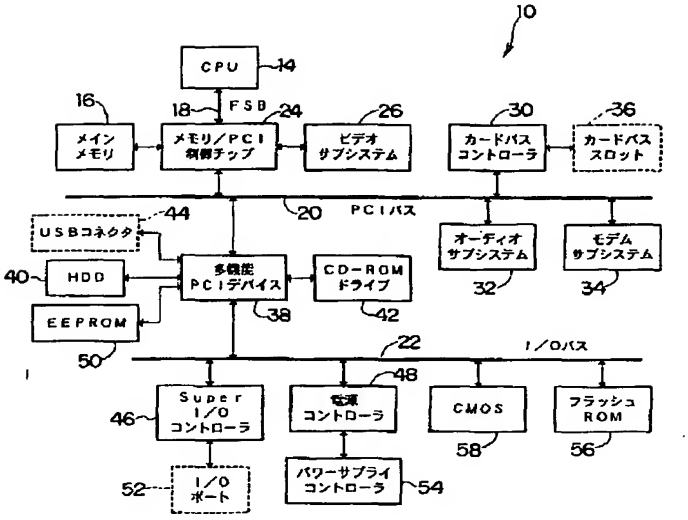
【図１１】　３種類以上のＯＳを順にブートする場合のブート・イメージのダウンロード及びブートの方法の一例を示すイメージ図である。

【図１２】　３種類以上のＯＳを順にブートする場合のブート・イメージのダウンロード及びブートの方法の他の例を示すイメージ図である。

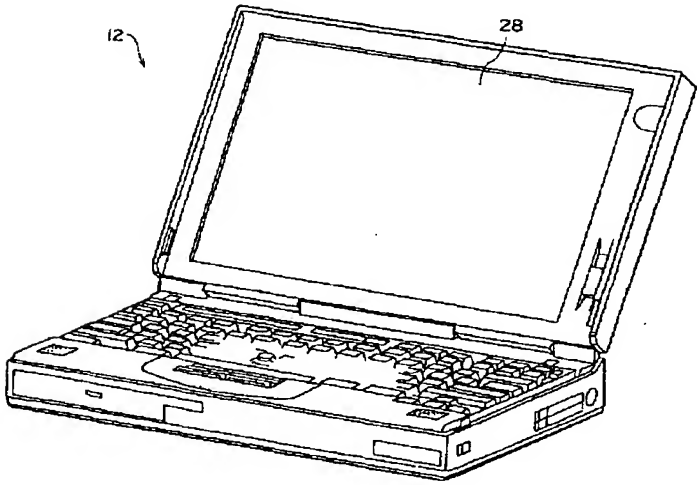
【符号の説明】

１０　コンピュータ・システム
１２　ＰＣ
１４　ＣＰＵ
１６　メイン・メモリ
４０　ＨＤＤ

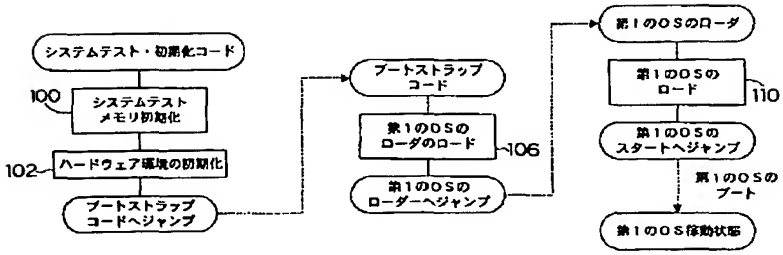
【図１】


10
CPU
14
16
18
FSB
24
26
30
36
メイン
メモリ
メモリ／PCI
制御チップ
ビデオ
サブシステム
カードバス
コントローラ
カードバス
スロット
20
PCIバス
USBコネクタ
44
オーディオ
サブシステム
モデム
サブシステム
40
HDD
多機能
PCIデバイス
CD-ROM
ドライブ
32
34
38
42
EEPROM
50
22
I／Oバス
48
Super
I／O
コントローラ
電源
コントローラ
CMOS
フラッシュ
ROM
46
58
56
52
I／O
ポート
パワーサプライ
コントローラ
54

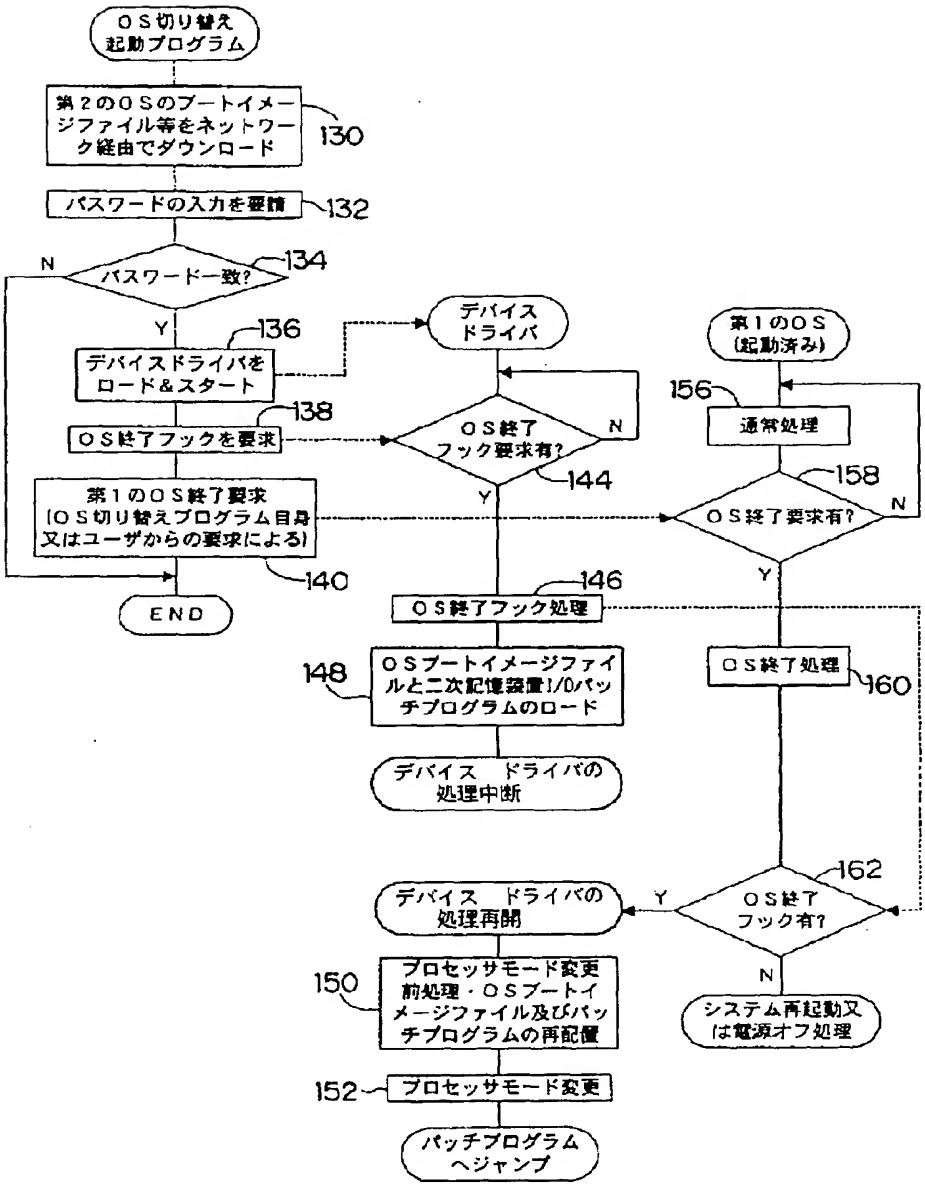
【図２】


12
28

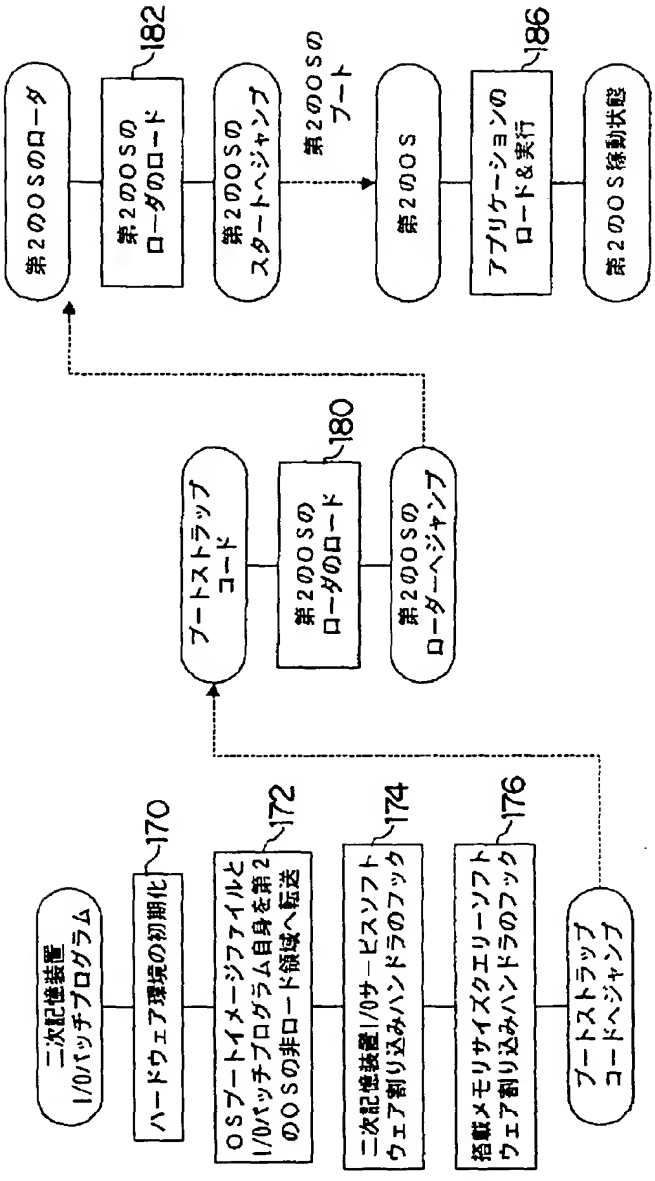
【図３】


システムテスト・初期化コード
100
システムテスト
メモリ初期化
102
ハードウェア環境の初期化
ブートストラップ
コードへジャンプ
ブートストラップ
コード
第１のOSの
ローダのロード
106
第１のOSの
ローダーへジャンプ
第１のOSのローダ
第１のOSの
ロード
110
第１のOSの
スタートへジャンプ
第１のOSの
ブート
第１のOS稼動状態

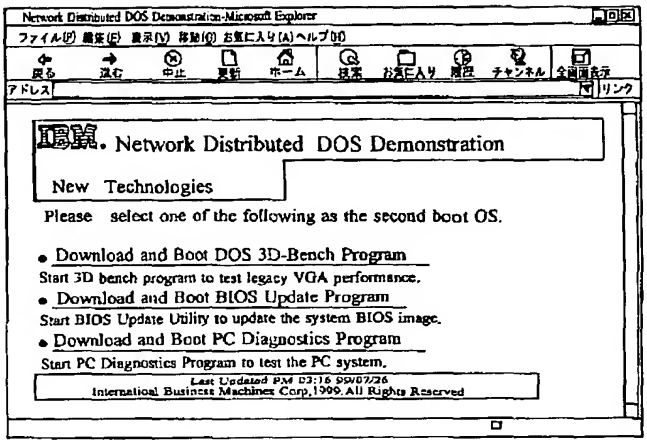
【図４】


ＯＳ切り替え
起動プログラム
第２のＯＳのブートイメージファイル等をネットワーク経由でダウンロード
130
パスワードの入力を要請
132
パスワード一致?
134
N
Y
デバイスドライバをロード＆スタート
136
ＯＳ終了フックを要求
138
第１のＯＳ終了要求
(ＯＳ切り替えプログラム自身又はユーザからの要求による)
140
END
デバイスドライバ
ＯＳ終了フック要求有?
144
N
Y
ＯＳ終了フック処理
146
ＯＳブートイメージファイルと二次記憶装置I/Oパッチプログラムのロード
148
デバイス　ドライバの処理中断
第１のＯＳ(起動済み)
156
通常処理
158
ＯＳ終了要求有?
N
Y
ＯＳ終了処理
160
ＯＳ終了フック有?
162
Y
N
デバイス　ドライバの処理再開
プロセッサモード変更前処理・ＯＳブートイメージファイル及びパッチプログラムの再配置
150
プロセッサモード変更
152
パッチプログラムへジャンプ
システム再起動又は電源オフ処理

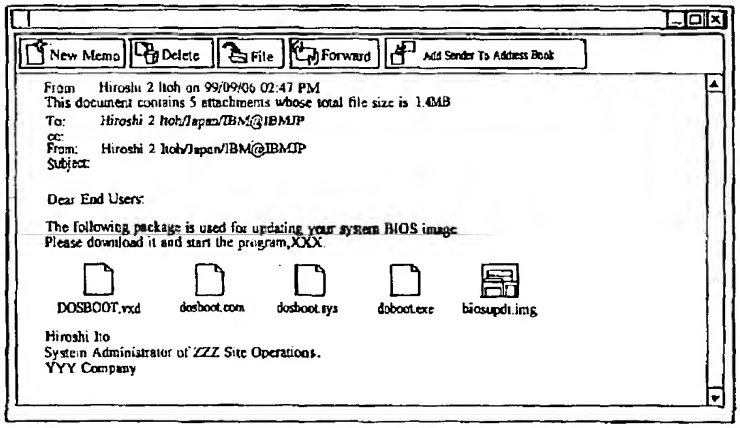
【図５】


二次記憶装置
I/Oバッチプログラム
ハードウェア環境の初期化
170
OSブートイメージファイルと
I/Oバッチプログラム自身を第2
のOSの非ロード領域へ転送
172
二次記憶装置I/Oサービスソフト
ウェア割り込みハンドラのフック
174
搭載メモリサイズクエリーソフト
ウェア割り込みハンドラのフック
176
ブートストラップ
コードへジャンプ
ブートストラップ
コード
第2のOSの
ローダのロード
180
第2のOSの
ローダへジャンプ
第2のOSのローダ
第2のOSの
ローダのロード
182
第2のOSの
スタートへジャンプ
第2のOSの
ブート
第2のOS
アプリケーションの
ロード&実行
186
第2のOS稼動状態

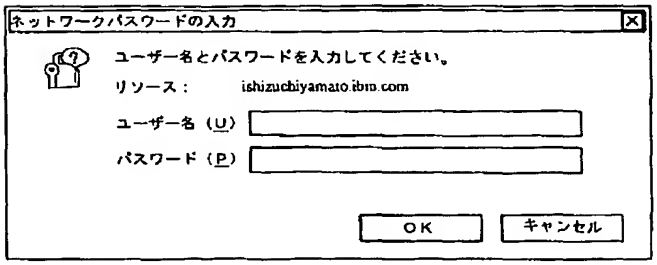
【図６】


Network Distributed DOS Demonstration-Microsoft Explorer
Network Distributed DOS Demonstration
New Technologies
Please select one of the following as the second boot OS.
• Download and Boot DOS 3D-Bench Program
Start 3D bench program to test legacy VGA performance.
• Download and Boot BIOS Update Program
Start BIOS Update Utility to update the system BIOS image.
• Download and Boot PC Diagnostics Program
Start PC Diagnostics Program to test the PC system.
International Business Machines Corp.1999. All Rights Reserved

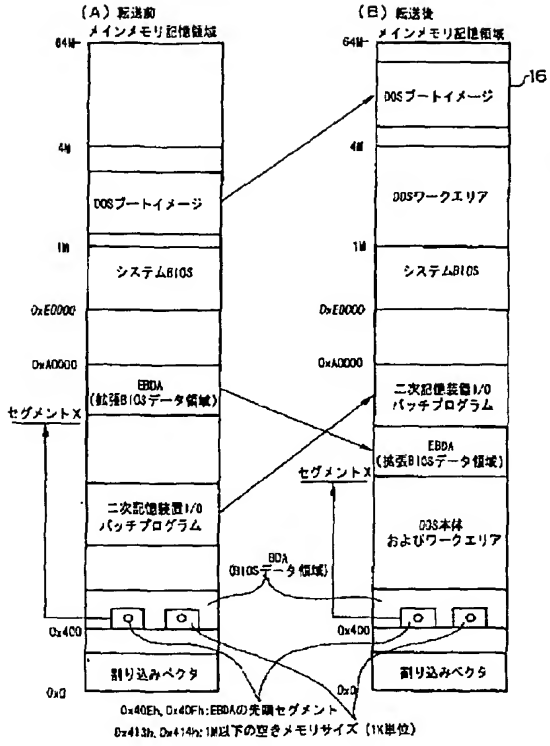
【図７】


New Memo
Delete
File
Forward
Add Sender To Address Book
From: Hiroshi 2 Itoh on 99/09/06 02:47 PM
This document contains 5 attachments whose total file size is 1.4MB
To: Hiroshi 2 Itoh/Japan/IBM@IBMJP
cc:
From: Hiroshi 2 Itoh/Japan/IBM@IBMJP
Subject:
Dear End Users:
The following package is used for updating your system BIOS image.
Please download it and start the program,XXX.
DOSBOOT.vxd
dosboot.com
dosboot.sys
doboot.exe
biosupdt.img
Hiroshi Ito
System Administrator of ZZZ Site Operations.
YYY Company

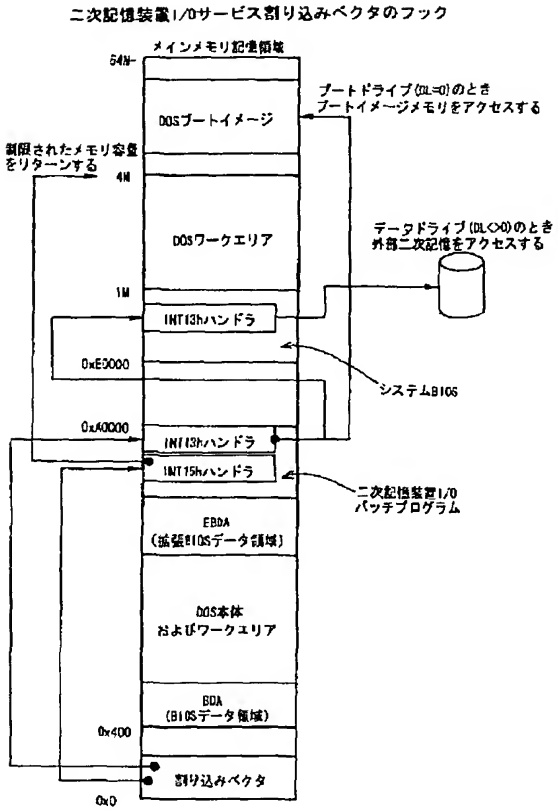
【図８】


ネットワークパスワードの入力
ユーザー名とパスワードを入力してください。
リソース： ishizuchiyamato.ibm.com
ユーザー名（U）
パスワード（P）
OK
キャンセル

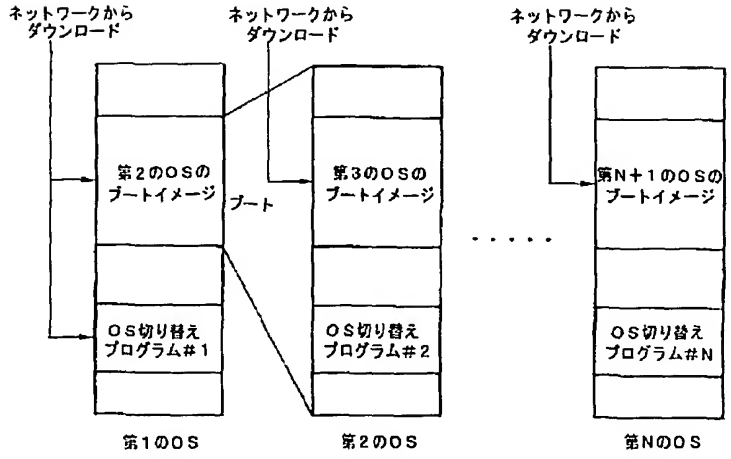
【図９】


（A）転送前
メインメモリ記憶領域
64M-
4M
1M
0xE0000
0xA0000
0x400
0x0
DOSブートイメージ
システムBIOS
EBDA
(拡張BIOSデータ領域)
セグメントX
二次記憶装置I/O
パッチプログラム
BDA
(BIOSデータ領域)
割り込みベクタ
（B）転送後
メインメモリ記憶領域
16
DOSブートイメージ
DOSワークエリア
システムBIOS
二次記憶装置I/O
パッチプログラム
EBDA
(拡張BIOSデータ領域)
DOS本体
およびワークエリア
0x40Eh, 0x40Fh:EBDAの先頭セグメント
0x413h, 0x414h:1M以下の空きメモリサイズ（1K単位）

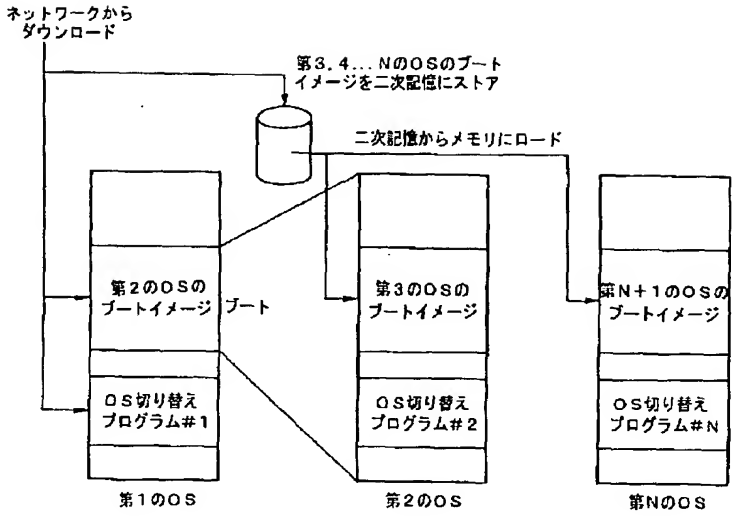
【図１０】


二次記憶装置I/Oサービス割り込みベクタのフック
メインメモリ記憶領域
64M-
4M
1M
0xE0000
0xA0000
0x400
0x0
DOSブートイメージ
DOSワークエリア
INT13hハンドラ
INT13hハンドラ
INT15hハンドラ
EBDA
(拡張BIOSデータ領域)
DOS本体
およびワークエリア
BDA
(BIOSデータ領域)
割り込みベクタ
制限されたメモリ容量
をリターンする
ブートドライブ(DL=0)のとき
ブートイメージメモリをアクセスする
データドライブ(DL<>0)のとき
外部二次記憶をアクセスする
システムBIOS
二次記憶装置I/O
パッチプログラム

【図１１】


ネットワークから
ダウンロード
ネットワークから
ダウンロード
ネットワークから
ダウンロード
第２のOSの
ブートイメージ
ブート
第３のOSの
ブートイメージ
第N＋１のOSの
ブートイメージ
OS切り替え
プログラム#１
OS切り替え
プログラム#２
OS切り替え
プログラム#N
第１のOS
第２のOS
第NのOS

【図１２】



フロントページの続き

(72)発明者　加藤　直孝
神奈川県大和市下鶴間1623番地14　日本アイ・ビー・エム株式会社　大和事業所内

Ｆターム(参考)　5B076 BA02 BB06
5B098 GA02 GC12